UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS ARARANGUÁ

Jeiel Miguel Lopes

UM ESTUDO COMPARATIVO ENTRE BANCOS DE DADOS CONSIDERANDO AS ABORDAGENS RELACIONAL E ORIENTADA A GRAFO

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação.
Orientador: Prof. Dr. Alexandre Leopoldo Gonçalves.

Araranguá
2014

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lopes, Jeiel Miguel
UM ESTUDO COMPARATIVO ENTRE BANCOS DE DADOS CONSIDERANDO AS ABORDAGENS RELACIONAL E ORIENTADA A GRAFO / Jeiel Miguel Lopes ; orientador, Alexandre Leopoldo Gonçalves - Araranguá, SC, 2014.
90 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá. Graduação em Tecnologias da Informação e Comunicação.

Inclui referências

1. Tecnologias da Informação e Comunicação. 2. Grafo. 3. Tecnologias de Banco de Dados. 4. Plataforma Lattes. 5. Gerenciamento de Informação. I. Gonçalves, Alexandre Leopoldo. II. Universidade Federal de Santa Catarina. Graduação em Tecnologias da Informação e Comunicação. III. Título.

Jeiel Miguel Lopes

UM ESTUDO COMPARATIVO ENTRE BANCOS DE DADOS CONSIDERANDO AS ABORDAGENS RELACIONAL E ORIENTADA A GRAFO

Esta Monografia foi julgada adequada para obtenção do Título de "Bacharel em Tecnologias da Informação e Comunicação", e aprovada em sua forma final pelo Curso de Graduação em Tecnologias da Informação e Comunicação.

Araranguá, 09 de Dezembro de 2014.

Prof. Vilson Gruber, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Alexandre Leopoldo Gonçalves, Dr.
(Orientador)

Prof. Juarez Bento da Silva, Dr.

Profa. Olga Yevseyeva, Dra.

Portanto, tudo que vós quereis que os homens vos façam, fazei-lo vós também a eles.

Jesus – Mateus 7.12

AGRADECIMENTOS

Sobre tudo sou grato ao meu Deus que me proporcionou esta oportunidade, que me deu saúde, guiou-me, deu-me uma família maravilhosa, envolveu-me de pessoas amadas e de amigos que se tornaram minha firme estrutura pra superar os obstáculos deste caminho.

Agradeço ao meu pai, Jaime, e à minha mãe, Benta, que não mediram esforços para que eu pudesse trilhar esta caminhada. Aos meus irmãos, que me apoiaram e depositaram fé neste trabalho.

Agradeço especialmente ao Orientador, Prof. Dr. Alexandre Leopoldo Gonçalves, que me acompanhou desde o primeiro dia em que ingressei nesta instituição de ensino. Que me serviu de guia e de espelho. Sou grato pela sua dedicação e pelas oportunidades proporcionadas que me possibilitaram amadurecer profissional e academicamente.

Às pessoas amadas, que me acolheram nos momentos de dificuldades, e que juntamente com meus amigos, compreenderam minha ausência durante o período de elaboração deste trabalho.

Sou grato aos gestores da Betha Sistemas® que compreenderam minhas limitações de horário e me apoiaram para que eu pudesse trilhar este caminho com entusiasmo e dedicação.

Por fim, sou grato pelas pessoas que me inspiraram ainda antes de meu ingresso na academia. À minha amada tia Antônia que muito "sonhou" por mim, aos entes queridos que comigo celebram e aos que partiram deixando saudades.

Por tudo sou grato.

RESUMO

Nos últimos anos, como reflexo do advento da tecnologia surgiu a necessidade de automatizar não somente a forma de persistir a informação, mas a capacidade de integrá-la de forma eficiente. Para atender esta demanda, por um longo período, o modelo relacional de bancos de dados se tornou predominante no mercado de *software*. Entretanto, por conta da expansão do conceito vindo da Web 2.0 em que novos dados são produzidos rapidamente, tem-se afirmado que o modelo relacional apresenta problemas no que tange a escalabilidade e tempos de resposta a consultas complexas. Visando atender a esta nova demanda a comunidade acadêmica passa a discutir novas soluções, entre elas o uso de modelos de bancos de dados orientados a grafo que permitam uma administração voltada não mais pela estrutura em si, mas com foco na informação e em seus relacionamentos. Os grafos podem ser aplicados em diversos cenários da vida real e sua representação em um modelo de banco de dados pode simplificar o gerenciamento da informação. Este trabalho vislumbra a utilização destes conceitos no domínio da Ciência Tecnologia e Inovação (CT&I), mais especificamente sobre a base de currículos da Plataforma Lattes. Para tal, foi elaborado um protótipo que após a integração de um conjunto de currículos em memória realiza a persistência nos dois modelos de dados objetos de estudo deste trabalho, o relacional e o orientado a grafo. De acordo com análises comparativas realizadas no cenário estabelecido para este trabalho em que há uma evolução temporal dos dados correlacionados, o banco de dados relacional apresentou desempenho superior quando considerada a individualidade da transação. Já o modelo orientado a grafo (através do banco de dados Neo4J) apresentou uma degradação no desempenho quando existe a necessidade de múltiplas e constantes escritas em disco. Por outro lado, quando os dados são mantidos em memória e a confirmação em disco é postergada, o Neo4j apresenta um desempenho superior na fase de correlação dos dados. Conclui-se que o modelo orientado a grafo tende a ser superior caso este possua disponibilidade de memória, enquanto que o relacional é mais indicado para aplicações que exijam acessos frequentes ao meio de armazenado físico.

Palavras-chave: Grafo; Banco de Dados; Plataforma Lattes, Gerenciamento de Informação.

ABSTRACT

In recent years with the advent of technology came the need to automate not only the way to persist the information, but the ability to integrate it efficiently. Relational databases have become predominant in the software market to meet this demand. However, due to the expansion of the concept from the Web 2.0 in which new data is produced quickly, it has been said that the relational model faces problems taking into account scalability and response times to complex queries. The academic community has discussed new solutions including the use of graph oriented models permitting an administration oriented no longer to the structure itself, but focused on information and in their relationships in order to meet this new demand. The graphs can be applied in many real-life scenarios and their representation in a database model can simplify the information management. This work presents the use of these concepts in the field of Science, Technology and Innovation (STI), more specifically on the basis of Lattes Platform. For this purpose, we designed a prototype that after the integration of a set of curricula in memory performs the persistence in both models, relational and graph oriented. According to comparative analyzes of the established scenario in this work in which there is a temporal evolution of correlated data, relational database showed better performance when considering the transaction aspect. The graph oriented model (via Neo4J database) underperforms when there is a need for multiple and constant disk writes. On the other hand, when the data are kept in memory and disk confirmation is postponed, the Neo4j shows better performance better in the correlation phase of data. We conclude that the model oriented graph is more efficient if it has available memory, while the relational is more suitable for applications that require frequent disk access.

Keywords: Graph; Database; Lattes Platform, Information Management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID - Atomicidade, Consistência, Isolamento, Durabilidade
CAP - Consistência; Disponibilidade; Distribuível
CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPU - Unidade Central de Processamento
CT&I - Ciência, Tecnologia e Inovação
DCL - *Data Control Language* (Linguagem de Controle de Dados)
DDL - *Data Definition Language* (Linguagem de Definição de Dados)
DML - *Data Manipulation Language* (Linguagem de Manipulação de Dados)
EUA - Estados Unidos da América
FC - Frequência Conjunta
FN - Forma Normal
FNN - Forma Não Normal
IBM - *International Business Machines*
JDBC - *Java Database Connectivity*
MER - Modelo de Entidade-Relacionamento
NoSQL - *Not only SQL*
PB - *petabytes*
SEQUEL - *Structured English QUEry Language*
SGBD - Sistemas de Gerenciamento de Banco de Dados
SGBDR - Sistemas de Gerenciamento de Banco de Dados Relacionais
SQL - *Structured Query Language*
SSD - *Solid Static Disc*
TB - *terabytes*
TBM - *Terminal Business Service*
TI - Tecnologia da Informação
XML - *eXtensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos o incremento na capacidade de processamento computacional, a conectividade e o armazenamento de informações vem permitindo um avanço nas soluções providas pela Tecnologia da Informação (TI). Contudo, o início da área de Banco de Dados teve suas bases e origens em soluções simples ou pouco práticas para problemas impactantes nas rotinas organizacionais (BARROS, 2010).

Considerando as áreas relacionadas ao processamento, ao armazenamento e a localização da informação, estas iniciam sua evolução no final da década de 1950. Neste período, Bar-Hillel desenvolveu sua teoria sobre a recuperação de informações (GRIER, 2012). De maneira concomitante as empresas começaram a reconhecer que os negócios empresariais de fato necessitavam de soluções para processar suas informações.

De acordo com Grad (2012), na década de 1960 e início dos anos 1970, foram desenvolvidos os Sistemas de Gerenciamento de Banco de Dados (SGBD) em rede e hierárquicos que atenderam a demanda do mercado de softwares de *mainfraime*, sendo fortemente utilizados e vendidos até os anos 1990 por algumas das maiores das empresas de software independentes, entre elas, a IBM® (*International Business Machines*).

Ainda na década de 1970, Edgar Frank Codd, publicou um estudo sobre uma nova abordagem envolvendo a aplicação e criação dos conceitos de álgebra relacional (CODD, 1970). Posteriormente, Codd juntamente com a IBM® desenvolveu um SGBD com enfoque no modelo relacional que foi largamente difundido por três grandes corporações, IBM®, Oracle® e Ingres®, cujos elementos básicos são as relações (tabelas), as quais são compostas de linhas (tuplas) e colunas (atributos). Soma-se a isto a contribuição do trabalho de Donald Chamberlin, fundador da linguagem de programação dos SGBDs Relacionais (SGBDR), o SQL (*Structured Query Language*) (GRAD, 2012).

Através da linguagem SQL é possível manipular bancos de dados de forma otimizada, sendo que os comandos desta linguagem podem ser classificados em três grandes categorias: as instruções DDL - *Data Definition Language* (Linguagem de Definição de Dados), DCL - *Data Control Language* (Linguagem de Controle de Dados); e DML - *Data Manipulation Language* (Linguagem de Manipulação de Dados) (RAMAKRISHNAN; GEHRKE, 2008). Os comandos de DCL são

utilizados para conceder as permissões aos usuários sobre os objetos no banco de dados; os comandos DDL para criação de procedimentos, funções, tabelas, ou ainda a manutenção destes; e os comandos DML para execução de consultas e manipulação de dados na base, tais como os comandos *insert*, *update*, *delete* e *select*.

O modelo de banco de dados desenvolvido por Codd marcou não somente o início de uma nova era, mas estabeleceu uma relação entre os aspectos de sistemas gerenciadores de bancos de dados relacionais e os conceitos gerais de um sistema de informação. De acordo com Grier (2012), os conceitos de Codd concentram-se em projetar bancos de dados relacionais eficazes e eficientes, como uma nova e poderosa forma de organizar as informações de um sistema de informação.

Para William Marchant (1955 apud GRIER, 2012, p.12), manter os dados armazenados de forma organizada é parte relevante no processo de recuperação de informações com impacto no dia a dia das organizações. Marchant defendia que um sistema de informação, apesar de representar o progresso e a automatização dos processos empresariais, não iria retirar as pessoas do mercado de trabalho e sim otimizar o tempo e as rotinas organizacionais.

Apesar de o modelo de sistema de informação ilustrado por Marchant ter iniciado uma grande corrida pela implementação de soluções que pudessem comprovar sua eficiência, Grier (2012) menciona que sua tese foi contestada por um grupo de pesquisadores do renomado *Battelle Memorial Institute* em Ohio nos EUA. Esse grupo afirmava que a tecnologia computacional existente não comportaria a automatização do processamento, portanto para que isto fosse possível, seria necessário desenvolver não somente melhores computadores, mas também melhores maneiras de estruturar os dados armazenados.

De acordo com Simon (1957 apud GRIER, 2012, p.13), o fluxo de informações para tomada de decisões pode ser abstraído e relacionado em diversas direções em uma organização e a medida em que as organizações se tornaram mais complexas, as informações se tornam mais complexas. Soma-se a isso a exigência dos líderes dessas organizações em possuir informações com um maior nível de detalhamento e precisão.

Tanto a complexidade quanto a exigência no que tange a informação tem se alterado rapidamente. Segundo Wu et al. (2014), nas últimas décadas, com o desenvolvimento da Web, o volume de dados gerados por aplicações, soluções, recursos e tudo o que se refere a sistemas computacionais cresceu de forma expressivamente acelerada,

de forma que uma fração equivalente a 90 por cento dos dados atualmente existentes no mundo foram produzidos nos últimos dois anos. De acordo com Oliveira (2013), segundo projeções da IBM®, atualmente cerca de 15 petabytes (PB) de dados são processados diariamente.

Em Wu et al. (2014) são apresentados vários exemplos de aumento expressivo na produção de dados. O Flicker® por exemplo, entre fevereiro e março de 2012 gerou o equivalente a 3.6 terabytes (TB), o que representa cerca de 1.8 milhões de fotos por dia. Outros casos que podem ser observados são os de Redes Sociais como o Facebook® e o Micro Blog Twitter®, cujos recursos permitem que pessoas estabeleçam diferentes relacionamentos entre si associando informações e interagindo com uma enorme massa de dados que está disponível e é constantemente alimentada e movimentada por milhões de pessoas. Por exemplo, em 4 de outubro de 2012, quando ocorreu o primeiro debate entre o Presidente Barack Obama (EUA) e o Governador Rommey foram gerados mais de 10 milhões de *tweets* em apenas duas horas.

Em cenários como este, em que se faz necessário o gerenciamento de grande volume de dados, a utilização de um SGBD relacional pode representar um problema, visto que este aumento expressivo de dados passa a refletir diretamente numa maior demanda por escalabilidade, e com base nesta demanda, passou-se a questionar a eficiência do modelo convencional.

Um modelo de uma aplicação Web que pode exigir tal escalabilidade e apresentar um aumento exponencial de informações, são as redes de relacionamentos sociais, em que diversos indivíduos interagem de forma distribuída, relacionando e multiplicando dados constantemente.

A implementação de bancos de dados capazes de atender à necessidade apresentada neste novo cenário, a Web, passa então a conflitar com os conceitos de Codd (GRIER, 2012, p.15). Ligações e conexões dos diversos nós distribuídos na *Internet* não são facilmente implementadas numa estrutura relacional. O modelo de Codd teria sido desenvolvido para uma estrutura que permitiria se concentrar em certos tipos de relacionamentos.

É justamente nesse ponto que o foco das soluções não relacionais passa a ser discutido. De acordo com Ramos e Nascimento (2012), os modelos de dados não relacionais NoSQL (*Not only SQL*) apesar de terem sido idealizados a partir do modelo de dados descrito por Carlos

Strozzi em 1998, cujo objetivo era apresentar um novo SGBD Relacional não orientado por SQL denominado NoSQL® (Non-SQL – SGBDR), representam teorias distintas. Strozzi (1998) afirmava que seu SGBDR se apresentava como uma solução rápida e portátil, cujos limites de escalabilidade não poderiam ser determinados.

O movimento mais atual de NoSQL segue esta tendência, buscando mudanças como o aumento da escalabilidade e flexibilidades dos bancos de dados. Ramos e Nascimento (2012) realizam uma discussão geral sobre as funcionalidades e novidades deste modelo. Para eles, apesar de o NoSQL entrar em conflito com muitos dos paradigmas do modelo relacional, ele se apresenta não como um substituto a este modelo, mas como uma alternativa a aplicações com foco na flexibilidade e escalabilidade.

Dentro desta visão pode-se observar que os modelos se comportam diferentemente em alguns aspectos, tais como a metodologia de organização e armazenamento da informação, a escalabilidade mediante o aumento expressivo dos dados, os tratamentos de redundância, as linguagens de manipulação, a homogeneidade ou heterogeneidade do modelo mediante a mudança de estruturas. Soma-se a isso o controle de operações ACID consideradas fundamentais no modelo relacional e opcionais ou inexistentes em alguns dos modelos NoSQL.

1.1 PROBLEMÁTICA

Os SGBDs relacionais são altamente difundidos e utilizados, pois facilitam a implementação, a manutenção e o controle de aplicações, além de oferecerem uma validação automatizada de características específicas de bancos de dados como garantias de integridade, recuperação de falhas, controle de transações, otimização de consultas, atomicidade e controle de concorrência. Por outro lado, considerando os novos cenários que exigem soluções diferenciadas e o aumento expressivo dos dados, este modelo passou a ser questionado.

De acordo com Oliveira (2013), nas últimas décadas o volume de dados associados a sistemas computacionais apresentou um ritmo acelerado de crescimento.

Gerenciar um volume tão grande de dados e cada vez mais inter-relacionado exige a aplicação de diferentes estratégias, em que a distribuição da informação a fim de obter maior escalabilidade possui um papel fundamental. Em cenários como este, realizar o gerenciamento

dos dados utilizando um modelo relacional centralizado pode se tornar problemático, pois existe uma certa complexidade na interconexão das informações que se encontram distribuídas o que interfere principalmente na escalabilidade de sistemas desta natureza.

Uma aplicação bastante atual em que se promove a necessidade de alta escalabilidade é a estrutura de uma rede social como o Facebook®, uma vez que através desta ferramenta armazena-se o fluxo de informações geradas por milhões de pessoas que se comunicam entre si. A partir disto, surgiu a necessidade de se buscar novas soluções. Nos últimos anos Bancos de Dados não Relacionais (também chamados de NoSQL) cresceram em utilização objetivando atender demandas de escalabilidade (LAKSHMAN, 2010; LAI, 2010).

Por conta dessas novas demandas as soluções não relacionais ganharam relevância na provisão de escalabilidade e capacidade de distribuir a informação. Contudo, Banco de Dados Relacionais ocupam lugar de destaque em ambientes operacionais em que características como integridade e atomicidade de transações são essenciais. Estudos como os de Hadjigeorgiou (2013) mostram que este tipo de banco de dados se sobressai em operações de que envolvam a eliminação de informação, por exemplo, em relação aos não relacionais, que desempenham melhor em operações de inserção de dados.

Entretanto, domínios de aplicação que envolvam o mapeamento de instâncias e o relacionamento entre as mesmas considerando questões temporais, assim como, o aumento constante no volumes de dados, constituem desafios tanto para modelos tradicionais como o relacional quanto para modelos mais atuais ditos não relacionais.

Sendo assim, tem-se a seguinte pergunta de pesquisa do trabalho "Os bancos de dados orientados a grafo são superiores a SGBDs relacionais considerando um contexto de alta demanda de informação envolvendo características de interconexão temporal de elementos, ou seja, a formação de redes temporais?".

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Realizar um estudo comparativo entre as abordagens de Banco de Dados Relacional e Orientado a Grafo considerando aspectos gerais de desempenho.

1.2.2 Objetivos Específicos

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- Realizar um levantamento bibliográfico nas áreas de Banco de Dados com foco principal nos modelos Relacional e NoSQL, com ênfase no modelo Orientado a Grafo;
- Adaptar um modelo relacional que permita a carga de determinado domínio de aplicação com característica temporal;
- Propor um modelo em formato NoSQL (Orientado a Grafo) como alternativa ao modelo relacional;
- Desenvolver um protótipo que realize, a partir de dados de um domínio escolhido, a carga dos modelos relacional e NoSQL (Orientado a Grafo);
- Realizar um estudo comparativo levando-se em consideração aspectos de desempenho de carga de dados em ambas abordagens, relacional e NoSQL (Orientado a Grafo).

1.3 METODOLOGIA

O trabalho será desenvolvido com base um uma pesquisa exploratória uma vez que objetiva proporcionar uma maior familiaridade com o problema (GIL, 2008). Quanto a natureza pode ser vista com uma pesquisa aplicada, visto que pretende produzir conhecimento prático voltado à solução de problemas específicos (SILVA; MENEZES, 2005). É também caracterizada como tecnológica, pois objetiva o avanço da tecnologia com a produção de algum artefato (neste trabalho um protótipo) como resultado.

A metodologia de desenvolvimento deste trabalho é dividida em cinco etapas:

Etapa 1: estabelecer um estudo com base em um referencial teórico e literário sobre as áreas de bancos de dados com foco nos modelos Relacional e NoSQL, mais especificamente no modelo orientado a grafos.

Etapa 2: projetar os modelos de dados relacional e orientado a grafo, capazes de compor, armazenar informações que representem a correlação entre termos em geral, por exemplo, palavras-chave de artigos em uma base científica.

Etapa 3: modelar e configurar os bancos de dados;

Etapa 4: desenvolver uma aplicação que sumarize os dados a partir de uma fonte de informação pública e que sirva de suporte para este trabalho.

Etapa 5: desenvolver um protótipo que realize a carga dos dados em um cenário que possibilite a interconexão desses dados de maneira temporal.

Etapa 6: realizar um estudo comparativo do desempenho de carga dos dados obtidos a partir de fontes de informação pública utilizando as abordagens Relacional e Orientada a Grafo.

1.4 ORGANIZAÇÃO DO TEXTO

O documento está dividido em três capítulos. No presente capítulo apresenta-se o projeto, sendo realizada uma contextualização do tema através da introdução dos SGBDRs e demonstrando que este modelo tem enfrentado desafios em cenários que exigem alta escalabilidade. Desta forma, apresenta-se o modelo NoSQL como uma possível solução para tal necessidade, compondo assim a descrição da problemática, bem como dos objetivos gerais e específicos.

No segundo capítulo é detalhado o modelo de dados NoSQL demonstrando as principais características que estes bancos apresentam e realizando um aprofundamento no modelo de dados NoSQL orientado a Grafo com ênfase no Neo4J® e suas aplicações. Além disso, para permitir a contextualização e evolução histórica dos bancos de dados é realizado uma introdução sobre SGBDRs.

O terceiro capítulo apresenta o cenário utilizado neste trabalho e a proposição dos modelos de dados, Relacional e NoSQL. Além disso, discute os resultados obtidos buscando responder a pergunta deste trabalho de acordo com as possibilidades de análise existentes.

Por fim, são expostas as considerações finais e os trabalhos futuros.

2. BANCOS DE DADOS

Ao utilizar um cartão de crédito em uma transação online, independente da natureza, muito provavelmente, a aplicação que está estabelecendo a interface com o usuário acessa um banco de dados relacional para fazer a autenticação deste (WADE; CHAMBERLIN, 2012).

A tecnologia de Banco de Dados faz parte dos mais diversos cenários atualmente. De maneira geral, sua evolução tem permitido que aplicações sejam capazes de lidar de maneira adequada com uma variedade de problemas que envolvam o armazenamento e a recuperação de dados como suporte a operações rotineiras ou voltadas a tomada de decisão.

Nas próximas seções serão apresentadas as tecnologias de banco de dados relacionais e banco de dados NoSQL, mais especificamente os orientados a grafo.

2.1 BANCOS DE DADOS RELACIONAIS

A história do surgimento e expansão dos Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDRs), segundo Grad (2012), é certamente uma das mais interessantes histórias relacionadas ao desenvolvimento de tecnologias computacionais para o desenvolvimento e manutenção de software.

De acordo com Wade e Chamberlin (2012), o movimento que originou a abordagem de bancos de dados baseados em esquemas relacionais ocorreu a partir da mudança de Edgar Frank Codd em 1967 para a cidade Nova York, quando então, passou a integrar a equipe de pesquisa da IBM® no Laboratório de Pesquisas em *San José*. Em 1969, Codd escreveu um relatório técnico na IBM® em que abordava a problemática da livre redundância, bem como a importância da consistência dos relacionamentos em bancos de dados de larga escala.

Darwen (2012) por outro lado, apesar de intitular Codd como um dos precursores e grandes idealizadores dos bancos de dados relacionais, e por consequência os Sistemas Gerenciadores de Bancos de Dados Relacionais, resgata que este conceito já era utilizado antes da idealização do modelo de dados relacional. Desde a década de 1960, a IBM implementava soluções baseadas em TBM (*Terminal Business Service*) que exercia função similar a estipulada aos SGBDRs, entretanto de forma mais limitada.

Os TBMs permitiam acessar apenas um arquivo por execução cujos registros necessitavam estar todos no mesmo formato, não sendo também possível a criação de ponteiros entre registros, necessidade que veio a ser atendida pelos SGBDRs através da possibilidade de criação de chaves estrangeiras. Ainda de acordo com o resgate histórico realizado por Darwen (2012), os Sistemas Gerenciadores de Bancos de Dados Relacionais passaram a reter uma série de responsabilidades que até então necessitavam da implementação de diversos métodos de alta complexidade, tais como, a derivação de tabela e a criação das visões dos dados armazenados para os usuários.

Em 1970, Codd, com base no estudo realizado em parceria com o Laboratório de Pesquisas da IBM®, apresentou o modelo de armazenamento de dados de forma relacional denominado "*A Relational Model of Data for Large Shared Data Banks*". A ideia, apesar de ter gerado uma grande movimentação no meio científico, somente quase dez anos depois passou a receber a aceitação para aplicação em sistemas de banco de dados de cunho comercial. Esta aceitação, segundo Wade e Chamberlin (2012), deu-se também pelo trabalho executado por uma pequena equipe de pesquisadores do Laboratório de Pesquisas da IBM® em *San Jose* entre os anos 1970 e 1979, com o intuito de comprovar na prática a viabilidade do modelo.

O argumento central da ideia proposta por Edgar Frank Codd estava baseada no que este determinava como princípio básico da informação: "Todas as informações em um banco de dados devem ser representadas apenas uma vez e de uma única maneira, ou seja, por valores informados em colunas que compõem as linhas de uma tabela" (WADE; CHAMBERLIN, 2012).

Para Wade e Chabelin (2012), a aplicação deste princípio fundamentaria toda uma abstração de detalhes físicos em um banco de dados, tais como, a indexação de conteúdo, a criação de ponteiro de dados e o estabelecimento de estruturas de metadados. Esta abordagem visava uma maior produtividade por parte de desenvolvedores de sistemas que passaram a utilizar uma única linguagem de consulta estruturada para recuperação e manipulação dos dados no banco, a *Structure Query Language* (SQL).

A utilização dessa linguagem tinha como foco a interpretação e tradução para uma linguagem de baixo nível, em que a incumbência dos controles de transações concorrentes transferia a responsabilidade do programador para o SGBDR. Este por sua vez, deveria garantir a existência de propriedades básicas inerentes a manipulação dos dados,

chamadas de ACID (Atomicidade, Consistência, Isolamento e Durabilidade) (WADE; CHABELIN, 2012).

Desde então, a proposta apresentada por Codd em 1970, não somente ajudou a projetar um novo Sistema Gerenciador de Banco de Dados no mercado, mas também, transformou os investimentos em pesquisas voltadas para os bancos de dados baseados no modelo relacional. A partir disso, companhias como IBM®, Ingres®, Oralce®, Informix® e Sybase® passaram a operar de forma efetiva nesta tecnologia a partir de 1980 (ROWE, 2012; PREGER, 2012; CAMPBELL-KELLY, 2012).

2.1.1 Modelo Relacional

O modelo de armazenamento de dados de forma relacional foi proposto por Codd (1970) com base no estudo realizado em parceria com o Laboratório de Pesquisas da IBM®.

Uma das grandes preocupações da pesquisa era a identificação de metodologias que permitissem a determinado usuário reconhecer conceitualmente como um banco de dados estava organizado internamente.

Dentro deste conceito um banco de dados deveria ser capaz de prover a informação de forma coerente à aplicação mesmo que os dados estivessem sendo manipulados, uma vez que se previa o acesso concorrente da informação, o aumento gradativo dos dados, o aumento da demanda por escalabilidade e consequentemente, de disponibilidade para execução de consultas e emissão de relatórios.

Para Codd, uma base de dados deveria estar também organizada de forma normalizada. Para tanto, através de um estudo matemático, foram estabelecidas premissas que conduziram a uma representação matricial desprovida de ponteiros, e deste modo evitando o estabelecimento de esquemas de endereçamento baseados em *hash*. Na essência o modelo deveria ser livre de índices ou listas de ordenação, ainda que a representação física do banco de dados necessite de tais estruturas para se atingir desempenho em suas operações de manipulação dos dados.

De acordo com a pesquisa, ainda seria de fundamental importância que um SGBDR fosse capaz de prover algumas características de segurança, a fim de garantir que houvesse atomicidade nas transações, isolamento entre as seções, consistência da informação persistida e durabilidade dos dados. Tais características passaram então

a serem referenciadas como ACID (A=Atomicidade, C=Consistência, I=Isolamento e D=Durabilidade).

Portanto, um modelo de dados relacional poderia ser representado basicamente pela criação de relações (tabelas), onde cada tabela deve conter atributos (colunas), sendo possível estabelecer uma restrição de integridade única por registro (linha/tupla) através da utilização de chaves primárias (SILBERSCHATZ et al., 2006). Pode-se ainda relacionar duas tabelas através do vínculo de uma chave estrangeira, representada por uma coluna aferente a chave primária da tabela de origem. Para que isto ocorra de forma padronizada, foram definidas as formas normais, estabelecendo metodologias de abstração de dados de um cenário real e aplicação no modelo relacional.

2.1.2 Transações em Bancos de Dados

A utilização de um SGBD como provedor de gerenciamento dos dados é com certeza uma maneira mais econômica e eficiente de gestão de informação permitindo ao usuário armazenar e recuperar uma diversidade de dados através de alguns conjuntos de instruções. Estes conjuntos de instruções que realizam leituras e gravações dos dados em um SGBD são denominados de transações (SUMATHI; ESAKKIRAJAN, 2007).

Uma transação representa então, um conjunto de instruções em um banco de dados sendo efetuado por um usuário ou aplicação e controlado pelo SGBD, de forma que este, ao final, seja confirmada (através de um comando *commit*) ou revogada (através de um comando *rollback*).

Em outras palavras, várias instruções de leitura e escrita podem ser submetidas ao banco de dados na mesma transação de um usuário (SILBERSCHATZ et al., 2006).

Para Sumathi e Esakkirajan (2007), o gerenciador de transações desempenha uma função fundamental em um SGDB, de forma que este é responsável por garantir a eficiência e consistência dos dados. Os autores ressaltam ainda que a execução parcial de uma transação pode deixar o banco de dados em um estado inconsistente.

Sob a perspectiva das definições das principais categorias de instruções que podem ser submetidas para um banco de dados, Elmasri e Navanthe (2010) abordam os conceitos de DDL, DML e DCL. Ao elaborar um conjunto de instruções, cabe ao desenvolvedor considerar princípios de otimização e organização de código, uma vez que a

utilização de um comando DDL dentro da transação conduz a confirmação de todos os comandos anteriormente submetidos.

Segundo Sumathi e Esakkirajan (2007), um usuário de banco de dados pode requisitar ao SGBD diversas transações simultâneas. No paradigma proposto ao modelo de dados relacional, a consistência dos dados é uma característica inerente ao modelo, desta forma, sempre que ocorrer uma falha em uma determinada operação, o SGBDR deve ser capaz de reverte-la.

Visando garantir o estado consistente do banco de dados e prover um melhor rendimento efetivo do SGBD, o gerenciador de transações necessita então agendar as operações e fornecer o caminho mais seguro para completar a tarefa.

Para manter os dados na fase de acesso simultâneo e prevendo falhas do sistema, o SGBD precisa assegurar quatro propriedades importantes. Estas propriedades são chamadas de propriedades ACID (SUMATHI; ESAKKIRAJAN, 2007; ELMASRI; NAVANTHE, 2010; SILBERSCHATZ et al., 2006), conforme abaixo especificadas:

1. Atomicidade: Todas as operações da transação são refletidas corretamente no banco de dados ou nenhuma delas;
2. Consistência: A execução de uma transação isolada (sem qualquer outra transação) mantém a consistência do banco de dados;
3. Isolamento: Embora várias transações possam ser executadas de maneira simultânea, duas transações T_i e T_j não sabem o que cada uma está executando;
4. Durabilidade: Depois que uma transação for completamente executada com sucesso, as informações modificadas por ela persistem no banco de dados.

2.1.3 Formas Normais

Para que um banco de dados possa ser considerado de acordo com modelo de dados relacional e normalizado, ele deve passar pelo menos por três etapas de validação/normalização (HEUSER, 2009), sendo:

1. Primeira Forma Normal (FN1): Deve-se eliminar os grupos de dados repetidos, criando uma nova relação para esta categoria de dados a ser identificada por uma chave primária;

2. Segunda Forma Normal (FN2): Se um conjunto de valores são os mesmos para vários registros, eles devem ser movidos para uma nova tabela, sendo que as duas relações devem ser vinculadas com uma chave estrangeira;
3. Terceira Forma Normal (FN3): Os campos que não dependem da chave primária da tabela devem ser removidos e, se necessário, colocados em outra tabela.

Apesar da representação de um modelo adequado à FN 3 ser considerado suficientemente estruturado, Sumathi e Esakkirajan (2007) abordam ainda o quarto e quinto nível de normalização, os denominado "*Boyce Normal Form*":

1. Quarta Forma Normal (FN4): Remover dependências de valores múltiplos. Para tal, um conjunto de tabelas deve estar normalizado de acordo com a Forma Normal Boyce-Codd (BCNF), bem como, de acordo com a FN3, garantindo que cada determinante deve ser uma chave candidata.
2. Quinta Forma Normal (FN5): Remove as anomalias restantes. Desta forma uma relação somente será considerada normalizada se esta não mais puder ser decomposta em "*n*" relações mais simples.

2.1.4 O Processo de Normalização

Para realizar a aplicação das formas normais em Bancos de Dados Relacionais faz-se necessário entender para que serve e como deve ocorrer o processo de normalização.

A definição deste processo realizada por Sumathi e Esakkirajan (2007) apresenta a normalização como um processo de organização dos dados no banco. Para tanto pode-se incluir tabelas, estabelecer relações entre elas (considerando as regras concebidas para proteger os dados) e tornar o banco de dados mais flexível eliminando dois fatores: redundância e dependência inconsistente. O autor resume afirmando que a normalização consiste em analisar as dependências funcionais entre os atributos buscando o estabelecimento de relações bem estruturadas.

A existência de dados redundantes representa não somente um desperdício de espaço em disco, mas cria problemas de manutenção (SUMATHI; ESAKKIRAJAN, 2007). Segundo os autores, se um determinado dado existe em mais de um lugar, mediante a manutenção

deste, ele deve ser alterado exatamente da mesma maneira em todos os locais, o que além de não ser seguro é dispendioso. As dependências inconsistentes, por sua vez, podem dificultar o acesso e a recuperação dos dados, ou seja, o caminho para localizar os dados pode estar faltando.

Para Heuser (2009) a normalização de um esquema de dados serve a dois objetivos principais: (a) reagrupar as informações de forma a eliminar redundâncias de dados que possam existir nos arquivos; e (b) reagrupar as informações de forma que se possa prover um Modelo de Entidade-Relacionamento (MER).

Para a aplicação das metodologias de formalização de bancos de dados relacionais, podem-se utilizar técnicas de elaboração de esquemas textuais, cuja notação é incompleta, contudo compacta. Esta abordagem é útil para casos simples, bem como para discussões sobre a estrutura geral do banco de dados, onde não se faz necessário descrever com maiores detalhes a problemática apresentada.

De modo geral, estruturas não relacionais, mais antigas e baseadas simplesmente em sistemas de arquivos não utilizavam o conceito de normalização. Sendo assim, uma vez aplicado o processo de normalização os arquivos passam a ser representados em um modelo integrado através de esquemas relacionais (HEUSER, 2009).

Uma relação bem estruturada contém um conjunto mínimo de redundância e permite a inserção, alteração e exclusão, sem erros ou inconsistências. A normalização é um processo formal para decidir quais atributos devem ser agrupadas em uma relação (SUMATHI; ESAKKIRAJAN, 2007).

O processo de normalização nos permite então minimizar o trabalho de inserção, atualização e exclusão, reduzindo as anomalias e ajudando na manutenção na consistência dos dados (SUMATHI; ESAKKIRAJAN, 2007, ELMASRI; NAVATHE, 2009). Deste modo, a normalização objetiva:

1. Garantir que a semântica dos atributos no esquema de dados seja clara;
2. Reduzir a informação redundante nas tuplas armazenando cada fato dentro do banco de dados apenas uma vez;
3. Reduzir os valores nulos nas tuplas;
4. Reprovar a possibilidade de gerar tuplas falsas;
5. Distribuir os dados de forma que seja possível realizar mudanças ou manutenções;

6. Evitar possíveis anomalias na atualização dos dados;
7. Facilitar a realização das restrições de dados (*Checks, Foreign Keys, Unique Keys, Primary Keys*, etc);
8. Evitar codificação desnecessária (controle manual e tratamento da informação via *trigger*).

O grau de normalização, no entanto, é definido por formas normais que são aplicadas em um nível crescente de normalização, sendo elas: primeira forma normal (1FN), segunda forma normal (2FN), terceira forma normal (3FN), quarta forma normal (4NF) e quinta forma normal (5NF) (SUMATHI; ESAKKIRAJAN, 2007).

Para cada forma normal existe um conjunto de condições a serem aplicadas sobre o esquema de dados que garantem certas propriedades relacionadas com a redundância e atualização. Um modelo de dados normalizado na 3FN é considerado bem estruturado.

2.1.5 Aplicabilidade de Formas Normais em Bancos Relacionais

a) Forma Não Normal (FNN):

Pode-se representar a Tabela 1 de forma escrita em FNN: Proj (CodProj, Tipo, Descricao,(CodEmp, Nome, Categoria, Salario, DataIni, TempoAl)). A tabela está descrida em forma não normal, onde se descreve o envolvimento dos funcionários por projeto considerando o tempo de alocação destes, a data de início da alocação, seus salários e categorias.

Tabela 1: Tabela Não Normal.

Código do Projeto	Tipo	Descrição	Empregado					
			Código do Empregado	Nome do Empregado	Categoria	Salário	Data Inicial de Alocação	Tempo de Alocação (dias)
LSC001	Novo Desenvolvimento	Sistema de Estoque	2146	João	A1	4	01/11/91	24
			3145	Sílvio	A2	4	02/10/91	24
			6126	José	B1	9	03/10/92	18
			1214	Ana	A2	4	04/10/92	18
			8191	Isabel	A1	4	01/11/92	12
PAG02	Manutenção	Sistema de RH	8192	Mário	A1	4	01/05/93	12
			4112	Ana	A2	4	04/01/91	24
			6126	José	B1	9	01/11/92	12

Fonte: Heuser (2009).

Para que estas informações sejam armazenadas de acordo com os requisitos do modelo relacional, é necessária a aplicação das regras de formalização (FN1, FN2 e FN3):

b) Primeira Forma Normal (FN1):

As tabelas abaixo (Tabela 2 e Tabela 3) foram criadas aplicando a primeira forma de normalização na Tabela 1, onde os campos que representam grupos de dados repetidos foram eliminados, criando assim, uma nova relação (Tabela 2). Essa relação é identificada por uma chave primária (Código do Projeto) que passa a representar este conjunto de dados na Tabela 3 como chave estrangeira.

Tabela 2: Tabela de Projetos

Projeto		
Código do Projeto	**Tipo**	**Descrição**
LSC001	Novo Desenvolvimento	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

Fonte: Heuser (2009).

Tabela 3: Relacionamento entre as Tabelas Projeto e Empregado.

ProjEmp						
Código do Projeto	**Código do Empregado**	**Nome do Empregado**	**Categoria do Empregado**	**Salário**	**Data Inicial**	**Tempo de Alocação (dias)**
LSC001	2146	João	A1	4	01/11/1991	24
LSC001	3145	Sílvio	A2	4	02/10/1991	24
LSC001	6126	José	B1	9	03/10/1992	18
LSC001	1214	Ana	A2	4	04/10/1992	18
LSC001	8191	Isabel	A1	4	01/11/1992	12
PAG02	8192	Mário	A1	4	01/05/1993	12
PAG02	4112	Ana	A2	4	04/01/1991	24
PAG02	6126	José	B1	9	01/11/1992	12

Fonte: Heuser (2009).

Após aplicar as regras de normalização da FN1 a representação das tabelas de forma escrita ficaria conforme a disposição abaixo exemplificada:

Projeto(Codigo do Projeto, Tipo, Descrição)
ProjEmp(CodProj, CodEmp, Nome, Categoria, Salario, DataIni, TempoAl)

c) Segunda Forma Normal (FN 2):

Para se aplicar a FN2 é necessário verificar se existem conjuntos de valores repetidos para vários registros. Neste caso eles devem ser movidos para uma nova tabela, sendo que as duas relações devem ser vinculadas com uma chave estrangeira. Na Tabela 3 os dados referentes a empregados (Nome, Categoria e Salário) estão redundantes

para os empregados que trabalham em mais de um projeto. Desta forma, a Tabela 4 foi criada para armazenar estes dados e a Tabela 3 foi modificada, dando origem à Tabela 5 em que as informações agrupadas ao cadastro de funcionários passam a ser relacionadas pelo seu identificador "Código do Empregado".

Tabela 4: Tabela de Empregados.

Empregado			
Código do Empregado	**Nome do Empregado**	**Categoria**	**Salário**
2146	João	A1	4
3145	Sílvio	A2	4
6126	José	B1	9
1214	Ana	A2	4
8191	Isabel	A1	4
8192	Mário	A1	4
4112	Ana	A2	4

Fonte: Adaptado de Heuser (2009).

Tabela 5: Relacionamento entre as Tabelas Projeto e Empregado.

ProjEmp			
Código do Projeto	**Código do Empregado**	**Data Inicial**	**Tempo de Alocação**
LSC001	2146	01/11/1991	24
LSC001	3145	02/10/1991	24
LSC001	6126	03/10/1992	18
LSC001	1214	04/10/1992	18
LSC001	8191	01/11/1992	12
PAG02	8192	01/05/1993	12
PAG02	4112	04/01/1991	24
PAG02	6126	01/11/1992	12

Fonte: Adaptado de Heuser (2009).

A representação da FN2 de forma textual pode ser representada como:

Proj(CodProj, Tipo, Descricao)
ProjEmp(CodProj, CodEmp, DataIni, TempAl)
Emp (CodEmp, Nome, Categoria, Salario)

d) Terceira Forma Normal (FN3):

Para que se possa transformar o conjunto de dados para a FN3 é necessário que este, além de estar na FN2, todas as colunas da tabela devem ser dependentes da chave primária, ou seja, não devem existir dependências funcionais transitivas ou indiretas, que ocorrem quando uma coluna não chave primária depende funcionalmente de outra coluna ou combinação de colunas não chave primária.

Desta forma os campos que não dependem da chave primária da tabela devem ser removidos e, se necessário, colocados em outra tabela, como por exemplo, a aplicação realizada na Categoria e Salário do empregado, originando assim às tabelas abaixo (Tabela 6; Tabela 7).

Tabela 6: Tabela de Categorias.

Categoria	
Categoria	**Salário**
A1	4
A2	4
B1	9

Fonte: Heuser (2009).

Tabela 7: Tabela de Empregados.

Empregado		
Código do Empregado	**Nome**	**Categoria**
2146	João	A1
3145	Sílvio	A2
6126	José	B1
1214	Ana	A2
8191	Isabel	A1
8192	Mário	A1
4112	Ana	A2

Fonte: Heuser (2009).

A representação da FN3 de forma textual pode ser representada como:

Proj (CodProj, Tipo, Descricao)
ProjEmp (CodProj, CodEmp, DataIni, TempAl)
Emp (CodEmp, Nome, Categoria)
Categoria (Categoria, Salario)

2.1.6 Representação Tabular de um Banco Relacional

Nesta notação, são listadas as tabelas e para cada tabela os nomes das colunas que a compõem (entre parênteses), sendo que as colunas que compõem a chave primária aparecem sublinhadas. De acordo com Heuser (2009), após a definição de cada tabela, aparecem as definições das chaves estrangeiras conforme exemplo abaixo:

1. Definição da tabela: *<nome de tabela> (<nome de coluna>1,<nome de coluna>2,... <nome de coluna>n)*
2. Definição das chaves estrangeiras*: (<nome de coluna CE>1, <nome de coluna CE>n) referencia <nome de tabela>*
3. Exemplo de Notação: *Empregado(CodEmp, Nome, CodDepto, CategFuncional, CPF) CodDeptoreferencia Departamento Departamento (CodDepto, Nome)*

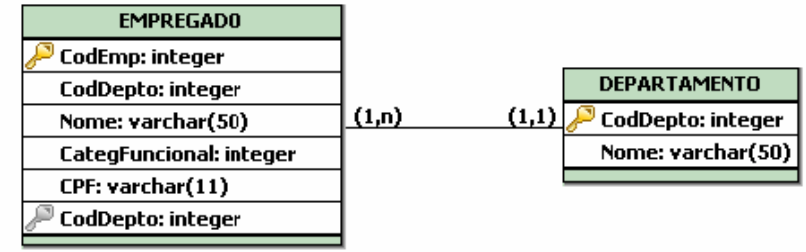
A representação de esquema de banco de dados relacional do exemplo acima ilustrado pode também ser apresentado na forma de esquemas diagramáticos (HEUSER, 2009) considerando que:

a) Cada tabela é representada por um retângulo;
b) As colunas que compõem a tabela são listadas dentro do retângulo representativo da tabela;
c) Deve-se também indicar o domínio de cada coluna (*INTEGER, VARCHAR(50)*);
d) As colunas que representam a chave primária devem estar identificadas de alguma maneira;
e) As linhas que conectam as tabelas indicam a existência de uma chave estrangeira. As colunas que compõem a(s) chave(s) estrangeira(s) devem ser identificadas.

A Figura 1 demonstra a aplicação das normas de elaboração de diagramas de bancos de dados relacionais, onde na tabela "EMPREGADO", a coluna "CodEmp" representa a chave primária da tabela, e a coluna "CodDepto" a chave estrangeira que relaciona o departamento do empregado com sua respectiva chave primária da tabela "DEPARTAMENTO". Para a chave estrangeira, ilustra-se uma ligação entre as tabelas, sendo que há uma indicação de cardinalidade para cada uma das relações.

Segundo Heuser (2009), o domínio das colunas por sua vez é descrito para cada atributo da tabela, de forma que, observando o diagrama se pode compreender que o "código do empregado" é um atributo do tipo número inteiro, enquanto seu "nome" é um campo de texto que pode ser composto por até cinquenta caracteres, e seu "CPF" um campo de texto com onze caracteres.

Figura 1: Esquema Diagramático.



Fonte: Adaptado de Heuser (2009).

A Tabela 8 apresenta a forma tabular do esquema diagramático da tabela de Empregado em que são indicadas algumas tuplas que preenchem as colunas respeitando o domínio (valores que podem preencher determinada coluna) de cada uma.

Tabela 8: Representação em forma tabular do esquema diagramático.

EMPREGADO				
Código Empregado	**Nome**	**Categoria Funcional**	**CPF**	**Depto**
1	Empregado A	1	278.486.262-37	RH
2	Empregado B	2	834.517.145-15	Financeiro
3	Empregado C	3	188.753.576-41	Comercial

Fonte: Adaptado de Heuser (2009).

2.1.7 SQL

A partir da implementação do modelo de dados relacional proposto por Codd em 1970 e a consequente necessidade de comprovar a viabilidade do modelo proposto, surge também a necessidade de uma linguagem capaz de abstrair a complexidade das transações realizadas e capaz de prover desempenho sobre as transações dos sistemas relacionais (WADE, 2012). De acordo com Wade (2012), Raymond Lorie, cientista da IBM, teria proposto o estabelecimento de comandos de criação, persistência e recuperação de dados que poderiam ser reutilizados de forma generalizada para o atendimento de requisições com valores específicos, surgindo então o SQL (*Struct Query Language*).

Segundo Elmasri e Navathe (2010), a linguagem SQL pode ser considerada um dos principais motivos de os bancos de dados relacionais terem atingido um patamar de sucesso. Uma característica importante para o estabelecimento desta linguagem é que ao se tornar um padrão, esta viabilizava a migração entre estruturas de diferentes SGBDs relacionais.

O nome SQL, conforme acima exposto significa Linguagem de Consulta Estruturada, contudo, inicialmente quando implementada pela IBM Researsh®, esta era chamada de SEQUEL (*Structured English QUEry Language*), criada para operar como uma *interface* para o experimento de um dos primeiros SGBDRs, o *SYSTEM R* (ELMASRI; NAVATHE, 2010).

A linguagem SQL possui três grupos principais de classificação de comandos. As instruções DDL - *Data Definition Language* (Linguagem de Definição de Dados), DCL - *Data Control Language* (Linguagem de Controle de Dados); e DML - *Data Manipulation Language* (Linguagem de Manipulação de Dados) (RAMAKRISHNAN; GEHRKE, 2008), podem ser submetidas numa única unidade lógica de trabalho, chamadas de transações.

Para Elmasri e Navathe (2010), os comandos da DCL são utilizados para conceder ou revogar (*grant/revoque*) permissões aos usuários de um banco de dados sobre os objetos contidos num esquema, seja para consulta, alteração, inserção, ou manipulação.

Os comandos da DDL são utilizados para criação, alteração ou exclusão de objetos, sejam estes procedimentos, funções, tabelas auxiliares, *triggers*, *checks.* Isto é obtido através da utilização dos comandos *create, alter* e *drop,* permitindo definir a estrutura do modelo físico do banco de dados.

Por sua vez, os comandos da DML são utilizados para execução de consultas e manipulação de dados na base, tais como, comandos de inserção de dados (*insert*), atualização (update), eliminação (delete) e consulta (select). Pode-se observar na Figura 2 um modelo em que são utilizadas as três classificações de comandos SQL para execução de uma determinada instrução no banco de dados:

Figura 2 Procedimento para atribuição de permissões.

```
if exists(select 1 from sys.sysprocedure where proc_name = 'premite_usuario'
          and creator = (select user_id from sys.sysuserperms
          where user_name = 'tecbth_conv')) then
    drop procedure tecbth_conv.premite_usuario;
end if
;
create procedure tecbth_conv.premite_usuario(ps_usuario varchar(20),ps_grant varchar(10))
begin
declare ls_comando long varchar;


llLoop: for ll as for_histCargos dynamic scroll cursor for
        select distinct tname as w_tabela from sys.syscolumns
            where creator = 'tecbth_conv' do
    set ls_comando = 'grant '||ps_grant||' on tecbth_conv.'||w_tabela||' to '||ps_usuario;
    print ls_comando ;
    execute immediate ls_comando ;
end for

end;

call tecbth_conv.premite_usuario('usuario_a','all');
call tecbth_conv.premite_usuario('usuario_b','select');
```

Fonte: Autor.

Na figura acima, o primeiro bloco de comandos representa a aplicação de uma consulta (*select,* DML) sobre o metadados do banco de dados relacional verificando a existência, ou não de um determinado procedimento. Caso a pesquisa retorne algum resultado, a instrução remete a execução de uma instrução de exclusão do objeto (*drop procedure,* DDL).

O segundo bloco inicia-se com a criação de um objeto do tipo procedimento (*create procedure*, DDL), cuja inexistência foi assegurada no bloco de comando anterior. No corpo do procedimento, uma nova consulta (*select*, DML) é submetida ao metadados para recuperar um grupo específico de tabelas voltadas a elaboração de uma instrução dinâmica de permissão (*grant*, DCL) sobre o objeto para o usuário definido no parâmetro. Por fim, as duas últimas instruções são invocações do procedimento informando via parâmetro o usuário que receberá as permissões, bem como o tipo de permissão a ser concedida.

2.2 NoSQL

De acordo com Vicknair et al. (2010), nos últimos anos os desenvolvedores de software tem estudado a utilização de sistemas de armazenamento de dados alternativos ao modelo convencional, ou seja, o modelo relacional. O NoSQL (*Not only* SQL) surge então como uma classificação geral para os diferentes tipos de SGBDs não relacionais.

O modelo de banco de dados dos bancos NoSQL tem por objetivo suprir algumas necessidades geradas pelo aumento exponencial da informação, principalmente em ambientes onde há elevada interatividade de usuários que compartilham informações entre si (RAMOS; NASCIMENTO, 2012).

Os primeiros projetos difundidos foram o Cassandra, BigTable, CouchDB, Project Voldemort e Dynamo. Para atender a demanda de armazenamento e gerenciamento de um alto volume de dados, estes bancos de dados desconsideravam algumas propriedades dos modelos relacionais, como por exemplo, as características ACID, permitindo também algumas características distintas aos SGBDRs abaixo listadas (VICKNAIR et al., 2010):

a) As tabelas podem conter colunas que existam em apenas alguns dos registros;
b) Cada atributo pode ser transformado em um objeto ou tabela;
c) Concepção de múltiplos relacionamentos;
d) Utilização de estruturas de árvore para armazenamento;
e) Possibilidade de constantes alterações no esquema de dados.

Para Neubauer (2010), o ganho que pode ser obtido pelo modelo NoSQL está dividido em quatro aspectos: a) leitura e escrita rápida; b) suporte para alto volume de dados; c) facilidade de expansão do banco de dados; e d) baixo custo. Contudo, para que fosse possível implementar estas propriedades muitos dos modelos propostos deixaram de ter suas operações associadas às características ACID e passaram a utilizar o Teorema de CAP (Consistência; Disponibilidade; Distribuível) que teria sido proposto pelo professor Eric Brewer em 2000. O Teorema de CAP (Figura 3) propunha que um sistema distribuído não pode atender a três necessidades distintas simultaneamente, mas só pode cumprir duas delas.

Figura 3: Teorema de CAP.

Fonte: Adaptado de Neubauer (2010).

De acordo com o Teorema de CAP apresentado na figura acima, as principais preocupações de alguns modelos NoSQL podem ser descritas conforme os apontamentos abaixo (HAN et al., 2011):

a) Consistência e Disponibilidade (*Consistency and Availability*): não há uma evidente preocupação em garantir que o banco de dados seja distribuído, mas em garantir a consistência do dado com alta disponibilidade, utilizando para isto, técnicas de replicação/redundância de dados. Este método pode ser encontrado em alguns gerenciadores de bancos de dados, tais quais Vertica (Orientado a Coluna), Aster Data e Greenplum (Multidimensionais com estrutura relacional).
b) Consistência e Distribuição (*Consistency and Partition Tolerance*): busca armazenar os dados em nós distribuídos, bem como garantir a consistência destes, contudo não garante a disponibilidade da informação. Os principais gerenciadores enquadrados neste modelo são: BigTable (Orientado a Coluna), Hypertable (Orientado a Coluna), HBase (Orientado a Coluna), MongoDB (Orientado a Documento), Terrastore (Orientado a Documento), Redis (Chave-Valor), Scalaris (Chave-Valor), MemcacheDB (Chave-Valor), Berkeley DB (Chave-Valor);

c) Disponibilidade e Distribuição (*Availability and Partition Tolerance*): esses sistemas garantem disponibilidade e distribuição do banco de dados procurando manter também a consistência. São exemplos destes bancos de dados: Voldemort (Chave-Valor), Tokyo Cabinet (Chave-Valor), KAI (Chave-Valor), CouchDB (Orientado a Documento), SimpleDB (Orientado a Documento), Riak (Orientado a Documento).

2.2.1 Modelos de NoSQL

Segundo Hecht e Jablonski (2011), os bancos de dados NoSQL podem ser divididos/classificados em quatro grupos:

a) Orientado a Chave – Valor (*Key Value Stores*): Este tipo de estrutura utiliza-se de uma chave única para endereçar um dado, sendo que uma vez que o pacote de dados é interpretado, as chaves são a única maneira de localizá-los no banco de dados. Este modelo normalmente é utilizado em operações simples que utilizam basicamente as chaves dos atributos, considerando para tanto o Project Voldemort®, Redis® e Membase®.

b) Orientado a Documentos (*Document Stores*): Neste modelo, os documentos são encapsulados em pares chave-valor únicos, sendo que cada documento passa a ter um identificador dentre toda a coleção de dados. Entre os gerenciadores encontram-se o CouthDB®, o MongoDB® e o Riak®.

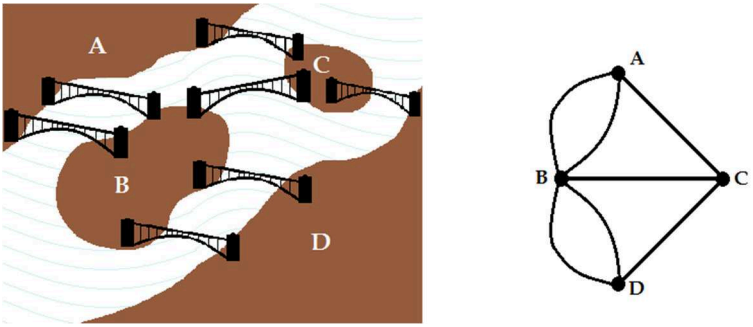
c) Orientado a Colunas (*Column Family Stores*): A estruturação de um banco de dados NoSQL orientado a colunas permite que os atributos possam ser agrupados em famílias de colunas, o que modifica a organização dos dados e possibilita um particionamento destes. Uma vez que as famílias de colunas estejam definidas, é possível adicionar novas colunas e linhas em tempo de execução. Um Modelo de banco de dados orientado a colunas é o Bigtable® da Google©, cujo sistema de armazenamento distribuído para gerenciar dados estruturados foi projetado com o intuito de lidar com grandes volumes de dados. Como alternativa para este modelo existem gerenciadores de código aberto, como o HBase® e o Hypertable®.

d) Orientado a Grafos (Graph Databases): Diferentemente dos modelos de dados relacionais, bem como dos demais modelos de Bancos de Dados denominados como NoSQL, os bancos de dados orientados a grafo objetivam um desempenho aprimorado na gestão eficiente de bases onde os dados são fortemente vinculados. Desta forma, o uso do grafo para este tipo de aplicação seria mais adequado, uma vez que o custo intensivo de operações recursivas ou junções podem ser substituído por eficiente recurso denominado de busca por referência cruzada (*traversal*) disponível em bancos de dados como o Neo4j® e oGraphDB®. Os bancos de dados orientados a grafo podem ser utilizados, por exemplo, em serviços baseados em localização geográfica, representação do conhecimento, sistemas de recomendação e em qualquer outra aplicação que se utilize de relações complexas, como o Twitter® que armazena um conjunto variado de relações entre usuários e suas publicações.

2.2.2 Teoria de Grafos

De acordo com Cardoso (2005), a origem da teoria dos grafos está associada ao problema das pontes de Königsberg (Alemanha), uma vez que a cidade localizava-se entre duas ilhas do rio Pregel as quais estavam ligadas umas das outras através de 7 pontes (Figura 4).

Figura 4: Pontes de Königsberg em 1736 e o respectivo grafo.



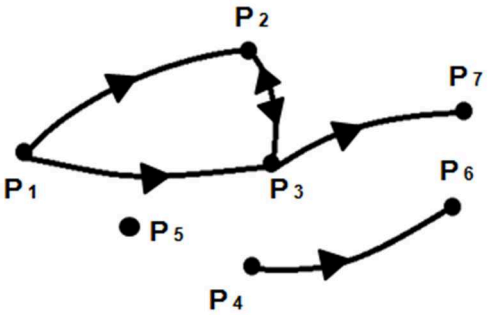
Fonte: Adaptado de Cardoso (2005).

A história transcreve que os habitantes de Königsberg tinham por hábito em seus passeios atravessar todas as pontes, contudo havia um certo desconforto pelo fato de que não ter sido encontrado uma forma de realizar um trajeto com partida e chegada a um mesmo lugar atravessando apenas uma vez cada uma das pontes (COSTA, 2013; ÁVILA; GROWNWALD, 2004).

Em meados de 1700, o matemático suíço Leonhard Euler, ao tomar conhecimento deste problema resolveu-o através da aplicação de uma estrutura chamada de grafo como apresentado na Figura 4 (CAVALCANTE; SILVA, 2009).

De acordo com Anton e Rorres (2012), grafos são conjuntos finitos de elementos {P1, P2, P3, ..., Pn}, bem como uma coleção finita de pares ordenados (Pi, Pj) de elementos distintos do conjunto, de forma que não haja repetição entre os pares ordenados. Os conjuntos são denominados "vértices", já os pares ordenados, "arestas". Um grafo pode ser matematicamente denotado como Pi→Pj (lê-se Pi está conectado a Pj), onde os vértices são representados por pontos, e as arestas por um segmento de reta direcionado, sendo que as arestas podem apontar tanto para um único sentido, quanto para os dois vértices conectados.

Figura 5: Exemplo de um grafo.



Fonte: Adaptado de Anton e Rorres (2012).

Os grafos podem ser divididos em duas grandes categorias: os direcionados denominados digrafo ou quiver, e os não direcionados, relacionados apenas como grafos.

Os grafos direcionados possuem um conjunto de vértices (V) e um conjunto de arestas (A), sendo s, t : A → V, onde "s(e)" é a fonte (*source*) e "t(e)" é o alvo (*target*) da aresta direcionada "e". Os grafos

não direcionados, por outro lado, são representados dado por um conjunto de vértices (V), um conjunto de arestas (A) e uma função w : A → P(V) que associa a cada aresta um subconjunto de dois ou de um elemento de V, interpretado como os pontos terminais da aresta (ANTON; RORRES, 2012).

O problema do caixeiro viajante pode representar a aplicação de pesos em um grafo ou dígrafo (GERSTING, 2001) onde cada caminho a ser percorrido é representado por uma aresta que recebe valor que denomina o seu "custo". Tendo estabelecido um grafo com pesos, o melhor caminho pode ser traçado através da aplicação de algoritmos.

Em 1956 o cientista da computação holandês Edsger Dijkstra, elaborou um algoritmo capaz de determinar o caminho mais curto em uma origem e todos os outros nós em um grafo com arestas de peso "não negativo" (HERNANDES et al., 2009). Assim, cada nó recebe um rótulo com um valor associativo correspondente a distância mais curta entre aquele nó e a origem. Contudo, a composição destes rótulos é temporária, pois a cada iteração, quando uma distância mais curta é encontrada, a alteração do rótulo passa a ser efetiva (KUROSE; ROSS, 2009).

2.2.3 Bancos de Dados Orientados a Grafo

Os Bancos de Dados Orientados a Grafo são sistemas gerenciadores que se utilizam de métodos de criação (c*reate*), leitura (*read*), atualização (*update*) e remoção (*delete*) (CRUD), para manutenção e manipulação de objetos em bases de dados através de sistemas de controle transacional e projetado de forma que a instância passe a ser trabalhada em memória (ROBINSON et al., 2013; HADJIGEORGIOU, 2013).

De acordo com Robinson et al. (2013) as tecnologias de Bancos de Dados Orientados a Grafo podem ser analisados considerando duas propriedades:

a) A forma de armazenamento: o armazenamento de dados pode ser realizado em forma de grafo (*native*) otimizado e projetado para o armazenamento e manutenção de grafos. Contudo, um Sistema Gerenciador Baseado em Grafo pode também serializar a persistência de dados do grafo em uma base relacional, orientada a objetos ou de uso geral fazendo a composição do grafo ao instanciar a base de dados em memória.

b) O mecanismo de processamento: o conceito de banco de dados orientado a grafo remete a existência de um ponteiro que vincula nós fisicamente conectados sem a utilização de índices de adjacência. Os autores consideram que quaisquer bancos de dados que trabalhem sob a perspectiva do usuário apresenta comportamento de um banco de dados orientado a grafo.

Com base neste conceito Robinson et al. (2013) apresenta diferentes Sistemas Gerenciadores de Bancos de Dados voltados para a arquitetura de Grafo, tais quais Microsoft Trinity®, Infinite Graph®, FlockDB®, Franz Inc.® , Allegro Graph®, Hypergraph DB®, Orient DB®, AffiilY®, *dex® e o Neo4j®.

Como pode-se observar na Figura 6, o Neo4j está listado entre os bancos de dados cujo mecanismo de processamento e forma de armazenamento são estabelecidos de forma nativa quando considerada a abordagem de grafo. Para a realização do estudo proposto neste trabalho que busca a comparação entre os paradigmas relacional e orientado a grafo, optou-se pela utilização deste banco de dados.

Figura 6: Arquitetura de um Banco de Dados Orientado a Grafo.

Fonte: Adaptado de Robinson et al. (2013).

2.2.3.1 Neo4J

O Neo4j é um Sistema Gerenciador de Banco de Dados capaz de proporcionar a manutenção de um banco de dados com diversas características. Segundo NEO4J (2014), entre as características destacam-se a robustez, a escalabilidade e o alto desempenho, sendo ainda capaz de garantir uma das mais importantes características dos bancos relacionais, as propriedades ACID. É mencionada ainda a capacidade de lidar com escalas de milhares de milhões de nós inter-relacionados. Outro aspecto mencionado é o desempenho em consultas, obtido através da utilização de "*traversals*" presentes em linguagens declarativas.

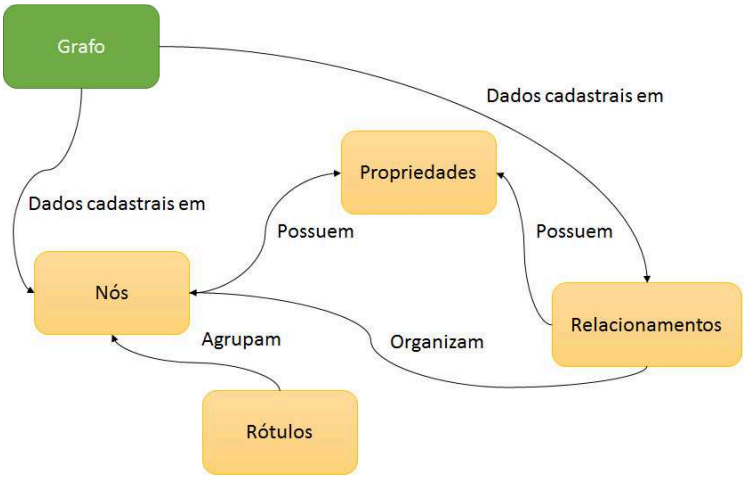
A aplicação das propriedades ACID é a base para se garantir a consistência dos dados. No Neo4j, todas as operações que modificam dados ocorrem dentro de uma transação, garantindo assim que os dados permaneçam em estado consistente (NEO4J, 2014). De acordo com Neubauer (2010), um grafo de bilhões de nós e relacionamentos pode ser tratado em uma única instância de um servidor. Contudo, quando a capacidade de transferência/manipulação de dados for insuficiente, o banco de dados de grafo pode ser distribuído entre vários servidores configurando um ambiente de alta disponibilidade.

O armazenamento de banco de dados orientado a grafo busca armazenar os dados estritamente conectados, de forma que, ao realizar a consulta através de "*traversals*" seja possível navegar milhões de nodos em frações de segundos (NEUBAUER, 2010).

De acordo com Abreu (2013), ao se utilizar bancos de bacos orientados a grafo é possível armazenar de forma simples e acessível quaisquer conjuntos de dados, uma vez que a representação de um grafo se assemelha a percepção que temos de uma situação de um problema do mundo real e estes são armazenados considerando o relacionamento dos objetos.

No Neo4J, para a composição de um grafo, é necessário realizar a adição de nós e de relacionamentos, onde os nós representam os vértices e os relacionamentos representam as arestas do conjunto (Figura 7) (NEO4J, 2014).

Figura 7: Representação da estrutura de um banco de dados orientado a grafos.

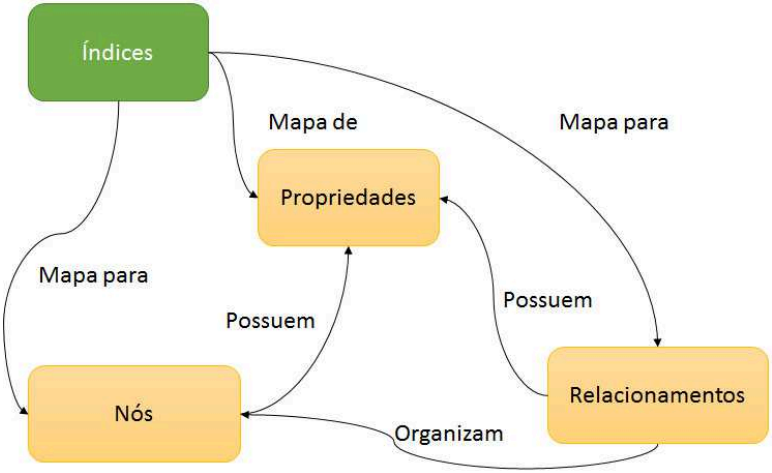


Fonte: Adaptado de Neo4j (2014).

A flexibilidade proporcionada por este modelo de armazenamento decorre do fato de que é possível adicionar propriedades distintas para o mesmo tipo de objeto (entende-se por propriedades em grafos como os atributos de uma relação do modelo relacional), tanto nos nós quanto nos relacionamentos. A recuperação destas informações pode ser realizada tanto pelo identificador do objeto, quanto por um conjunto de valores indexados por atributos específicos (ROBINSON et al., 2013).

Os índices podem ser criados tanto para os nós como para os relacionamentos, sendo que este pode auxiliar em determinada busca de um objeto (nó) inicial para a navegação do grafo através dos parâmetros repassados para a consulta (Figura 8).

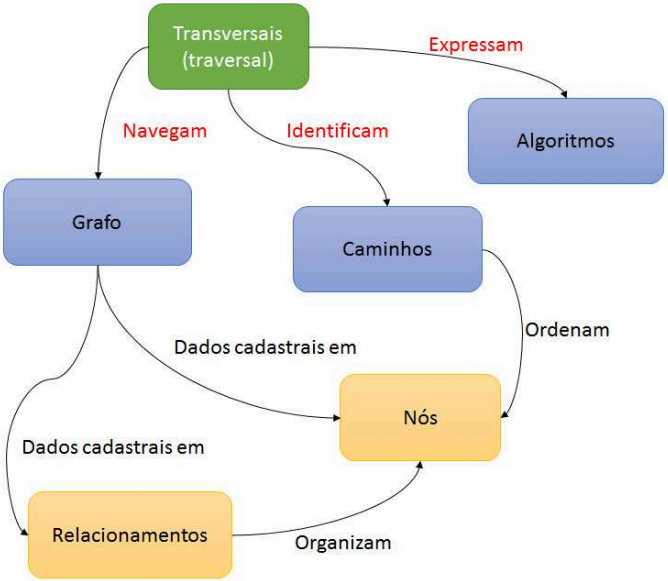
Figura 8: Representação da indexação de relacionamentos ou nodos por suas propriedades.



Fonte: Adaptado de Neo4j (2014).

A utilização de *traversals* é uma maneira otimizada de se navegar pelo grafo a partir de um ponto inicial deslocando-se até seus nós relacionados de acordo com um determinado algoritmo com o objetivo de encontrar respostas para perguntas estabelecidas. Para tanto, é possível estipular o limite da navegação para que esta execute até um determinado nível (N1, N2, ..., Nn), ou ainda, parametrizar a consulta para que esta se desloque até o término do grafo (Figura 9) (NEO4J, 2014; NEUBAUER, 2010).

Figura 9: Representação da navegação em um grafo utilizando *traversal*.

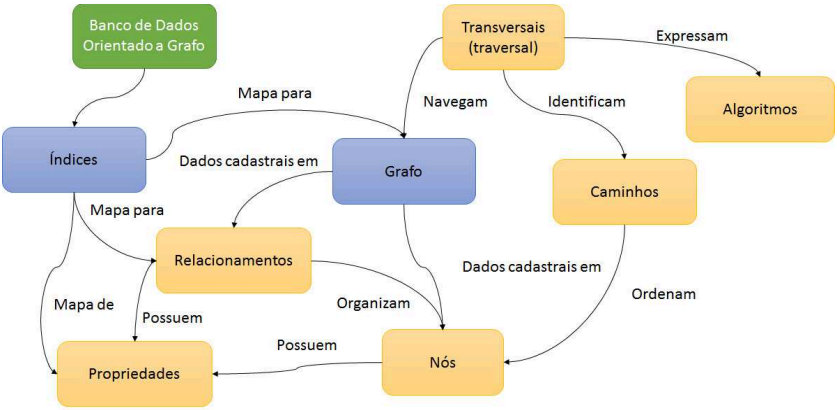


Fonte: Adaptado de Neo4j (2014).

Ainda de acordo com sua documentação, o Neo4j é um banco de dados orientado a grafo de código aberto suportado comercialmente, cujo projeto foi elaborado visando a confiabilidade e a otimização na manipulação de estruturas em grafo em vez de tabelas. Desse modo, uma aplicação trabalhando com Neo4j pode ter a expressividade de um grafo com a confiabilidade esperada de um banco de dados relacional.

Na Figura 10 é possível observar o funcionamento generalizado do Neo4j e de seus referidos objetos, sendo que através da utilização de *traversals* é possível navegar um grafo identificando caminhos onde os nós estão ordenados e organizados através de relacionamentos. A partir deste processo de navegação, é possível recuperar os dados armazenados em propriedades de nós e de relacionamentos.

Figura 10: Visão consolidada do funcionamento do Neo4J.



Fonte: Adaptado de Neo4j (2014).

Além dos recursos acima apresentados, este banco de dados oferece suporte a uma elevada gama de linguagens de programação e *frameworks* para manipulação e recuperação de dados, entre elas, pode-se mencionar as mais difundidas, como por exemplo, o *framework* Lucene e linguagem de consulta Cypher utilizadas para manutenção das bases através de uma API Java® (NEO4J, 2014; NEUBAUER, 2010).

2.3 DESAFIOS E TENDÊNCIAS EM BANCO DE DADOS

De acordo com as projeções da IBM®, cerca de 15 PB de dados estruturados e não estruturados são gerados diariamente (OLIVEIRA; 2013), e a capacidade de gerenciar este volume de dados, de forma que, se torne possível a extração e a composição de informações úteis é um desafio às empresas de tecnologia que visam entregar esta informação processada como um produto chave para obtenção de uma determinada vantagem de mercado (ASSUNÇÃO et al., 2013).

Schomm et al. (2013) declara que a elaboração de soluções para análises de informações com potencial para a tomada de decisão, sejam estruturados ou não, pode representar uma importante ajuda às corporações, propiciando uma visão não somente dos dados privados de determinada organização, mas de todo um contexto que pode ser obtido de publicações realizadas na Web, blogs ou redes sociais. Deste modo, agregam-se informações correlacionadas sobre as preferências de produtos e serviços de consumidores, por exemplo, em uma base

estratégica e de conhecimento que permita identificar e prever demandas.

Para Hashem et al. (2015), os desafios e questões futuras sobre a aplicações para gerenciamento de grandes volumes de dados em nuvem necessitam ser abordados, estudados e discutido pela academia e pela indústria, de forma que, pesquisadores, profissionais e estudiosos das ciências sociais possam também colaborar para garantir o sucesso a longo prazo da gestão de dados em um ambiente de computação em nuvem. Desta forma, o futuro das tecnologias de bancos de dados como solução para o expressivo crescimento no volume dos dados (*Big Data*), pode não somente proporcionar soluções capazes de agregar valor às organizações, mas também apresentar um número considerável de desafios principalmente na elaboração de uma infraestrutura diferenciada para o armazenamento, gerenciamento, interoperabilidade, governança e análise dos dados.

Davenport et al. (2010 apud ASSUNÇÃO et al., 2013) apresenta soluções aplicadas sobre o conceito de *Big Data* como ferramentas capazes de proporcionar aos tomadores de decisões, através da aplicação de técnicas de Mineração de Dados (*Data Mining)*, diretrizes que auxiliem na precisão de comportamentos futuros de novos tópicos/assuntos de interesse.

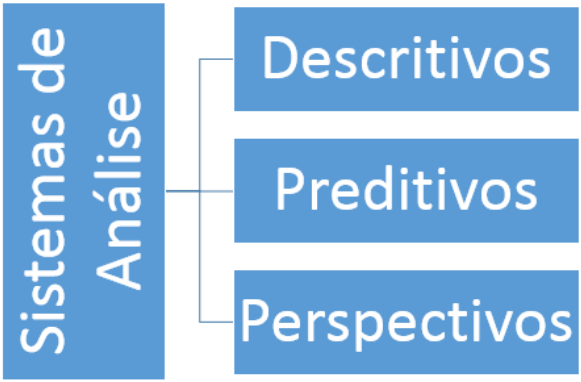
Apesar de não ser uma área nova, a Mineração de Dados vem ganhando destaque no cenário de grandes volumes de dados. Em Wu et al. (2014) os autores citam os desafios de *Data Mining* no contexto de *Big Data*. Em essência a Mineração de Dados tem como objetivo descobrir as inter-relações entre os atributos previamente desconhecidas e aparentemente não relacionados através da aplicação de métodos e heurísticas sobre conjuntos de dados que podem estar representados em diferentes formatos e estruturas (WITTEN et al., 2011).

A aplicação de técnicas e algoritmos de Mineração de Dados no contexto de *Big Data* permite compor um conjunto de dados cuja origem está distribuída, seja ela em bancos de dados, *streams*, *Data Warehouses* entre outros modelos, normalizados ou não. A composição deste grande volume de informação pode demandar diversas tarefas de pré-processamento para integração, adequação e filtragem, normalmente realizadas por aplicações e tecnologias que podem ser aplicadas em cenários de *Big Data*, tais como MapReduce, Dryad, Pregel, PigLatin, MangoDB, Hbase, SimpleDB e Cassandra (HASHEM et al., 2015).

Segundo Assunção et al. (2013), estes sistemas de análise podem ser divididos em três principais categorias, sendo, os descritivos, os

preditivos e os perspectivos (Figura 11). Os descritivos se utilizam de dados históricos para composição de padrões e elaboração de relatórios gerenciais modelando comportamentos passados. Os preditivos aplicam algoritmos de análise sobre bases históricas com foco em resultar a tendência futura. Os perspectivos por sua vez, buscam analisar os diferentes impactos que podem ser causados por um determinado grupo de ações a ser ou não tomado.

Figura 11: Categorias dos sistemas de análise.



Fonte: Adaptado de Assunção et al. (2013).

De modo geral, o desempenho dos sistemas de análise sobre grandes volumes de dados requer a aplicação de métodos eficientes para armazenamento, filtro, transformação e apresentação dos dados, e neste ponto, a utilização dos recursos em nuvem (*on Cloud*) se apresentam como um modelo repleto de desafios e oportunidades (ABADI, 2009; SAKR et al., 2011; KATZ, 2012).

O ambiente *on Cloud* pode, além das categorias anteriormente mencionadas, ser categorizado em três diferentes grupos de infraestrutura (ASSUNÇÃO et al., 2013):

a) Privado: é implantado em uma rede privada e gerido pela própria instituição ou por uma parceiro/empresa terceirizada, sendo aplicável para negócios onde se faz necessário um alto nível de controle de segurança e privacidade, ou para compartilhar os serviços e dados de forma mais eficiente nos diferentes departamentos de uma grande empresa.
b) Público: é implantado e aberto à Internet onde pode ser acessado por um público em geral, de forma que se possa ter

um alto índice de compartilhamento de informação com baixo custo.

c) Hibrido: combina as características dos modelos Privado e Público, onde se faz necessário restringir o acesso em meio público.

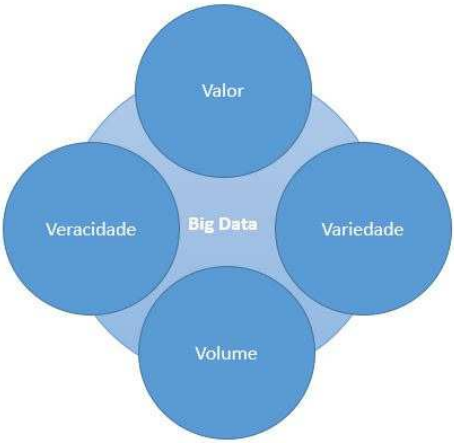
Krishna e Varma (2012) consideram que as implantações de sistemas de análise *on Cloud*, estão também relacionadas aos seguintes cenários no que se refere à disponibilidade de dados e modelos de análise:

a) Privado: os dados e o modelo são privados;
b) Público: os dados e o modelo são públicos;
c) Híbridos: os dados são públicos e o modelo é privado, ou, os dados são privados e os modelos são públicos.

Outra das principais características do conceito de *Big Data* é a composição do modelo multi-v (Figura 12) apresentando algumas características que ajudam a definir o grau de segurança e relevância gerado, armazenado e/ou processado (RUSSOM, 2011; ZIKOPOULOS et al., 2011; HASHEM et al., 2015; IBM, 2013):

a) Variedade: representa as diferentes entradas dos diferentes tipos de dados processados;
b) Velocidade: representa o espaço temporal necessário para produção e processamento dos dados, ou seja, o processamento dos dados deve ser realizado em tempo hábil, muitas vezes em tempo real;
c) Volume: representa a quantidade massiva de dados armazenados;
d) Veracidade: representa o quanto a informação pode ser considerada correta, considerando o grau de confiabilidade da fonte.

Figura 12: Os 4 Vs do Big Data.



Fonte: Adaptado de Hashem et al. (2015).

Não obstante, a crescente evolução dos recursos tecnológicos necessários para realizar a aplicação destas tecnologias com o intuito de entregar informação com alto valor agregado para a tomada de decisão ainda encontrará alguns desafios que necessitaram de análises específicas (ASSUNÇÃO et al., 2013), entre eles:

a) Variedade: como lidar com um volume crescente de dados, considerando que estes podem ser estruturados, não estruturados, semiestruturados ou mistos, de forma que seja possível extrair rapidamente um conteúdo significativo?
b) Armazenamento: como identificar e armazenar informações relevantes de forma eficiente considerando os conceitos de variedade e velocidade? Como armazenar informações de modo que estas sejam facilmente migradas/portadas entre provedores de serviços para computação de alta demanda?
c) Integração: como estabelecer mecanismos que se utilizem dos novos protocolos e *interfaces* para integração de dados visando a gerência de dados de natureza e fontes diversas?
d) Processamento de Dados e Gestão de Recursos: como utilizar os novos modelos de programação otimizados para *streaming* e/ou dados multidimensionais? Como otimizar o uso de recursos e consumo de energia durante a execução do aplicativo analítico?

A aplicação destes recursos fornece uma série de soluções frente aos desafios apresentados pelo cenário de crescimento do volume de informação. Contudo, estas soluções ainda estão limitadas a diferentes recursos e metodologias que nos últimos anos tem passado por uma grande evolução (HASHEM et al., 2015; ASSUNÇÃO et al., 2013; OLIVEIRA, 2013).

3 DESENVOLVIMENTO E AVALIAÇÃO DO TRABALHO

Considerando a problemática apresentada, este trabalho propõe a implementação de uma aplicação, nível de protótipo, para a realização de um estudo comparativo entre as abordagens de Banco de Dados Relacional e Orientado a Grafo levando em consideração os aspectos gerais de desempenho de cada um dos modelos.

Para a elaboração da comparação e obtenção de dados quantitativos, propõem-se neste capítulo dois modelos de representação dos dados, sendo o primeiro relacional e o segundo como uma alternativa ao relacional, o modelo NoSQL orientado a Grafo.

A aplicação implementada realiza uma carga de dados extraída de arquivos em formato XML que representam Currículos Lattes processando as frequências (ocorrências) conjuntas obtidas a partir das palavras-chave constantes em itens de produção científica e tecnológica. As informações obtidas a partir dos currículos são sumarizadas e utilizadas para popular ambos os modelos com o intuito de realizar um estudo comparativo avaliando-se o desempenho na etapa de carga.

Para o estudo utilizou-se um computador pessoal com processador Intel® Core i7-3632QM 2.2GHz com Turbo Boost de 3.2GHz; 6GB DDR3 de memória; disco SATA 3Gb/s, com 5400 rpm, disco (HDD) com capacidade de armazenamento de 500 GB com buffer de 16 MB; e Sistema Operacional Windows 8.1 Single Language© 2013 (Microsoft Corporation®).

3.1 APRESENTAÇÃO DO CENÁRIO

O gerenciamento de bases de dados nas quais os dados são alimentados de forma distribuída com a tendência de evolução temporal constitui um cenário desafiador quando existe a necessidade de integração dessa informação. A base de Currículos Lattes integrante da Plataforma Lattes se enquadra nessa característica. Em função da sua consolidação como insumo nos mais diversos processos que envolvam a análise da produção científica e tecnológica e da qualificação acadêmica/profissional, esta base de dados vem crescendo em volume de maneira considerável nos últimos anos.

A Plataforma Lattes é uma ferramenta elaborada pelo CNPq que possibilita a integração das informações de bases de dados de Currículos, Grupos de Pesquisa e Instituições através da aplicação de

um conjunto de sistemas computacionais (CNPq, 2014b). No contexto deste trabalho o foco reside na manipulação de dados curriculares, mas especificamente os dados do Currículo Lattes.

De acordo com Silva (2013), o objetivo do Lattes é o contínuo aprimoramento da qualidade das informações e racionalização das informações dos trabalhos de diversos atores de CT&I (Ciência, Tecnologia e Inovação) do Brasil.

A partir de seu objetivo de aprimoramento e da geração de informações sobre as produções acadêmicas, o modelo de armazenamento de dados do Currículo Lattes é uma estrutura robusta e extensa, na qual se persiste uma grande quantidade de informações sobre alunos, pesquisadores, instituições entre outros (SILVA, 2013).

Segundo Bianco (2011), o modelo de dados do Lattes está divido em três áreas específicas:

- Informações gerais composta por identificação pessoal, endereço profissional e residencial, formação e titulação acadêmica, experiência profissional, idiomas, premiações e titulações, entre outros;
- Produção científica e tecnológica envolvendo a produção bibliográfica, produção técnica e outras produções artísticas/culturais;
- Informações complementares incluindo: (i) formação complementar; (ii) participação em banca de trabalhos de conclusão; (iii) participações em eventos, congressos e outros; (iv) participações em bancas de comissões julgadoras; e (v) orientações em andamento.

O Currículo Lattes é composto por conjuntos de dados que correspondem às informações curriculares, atuais e passadas, que devem ser mantidas e atualizadas pelos próprios pesquisadores de forma que através da população destes dados, o CNPq possa extrair informações necessárias para fomentar a pesquisa e a extensão no país.

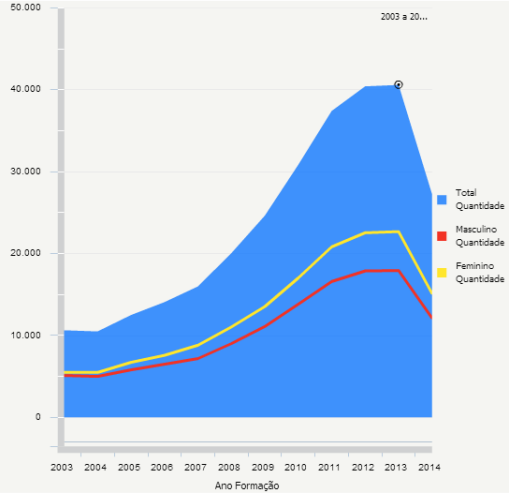
A base de dados Lattes constitui-se em uma importante ferramenta para elaboração de políticas públicas de investimento e análise da Ciência e Tecnologia. Isso é possível pela interconexão entre as informações, seja de pesquisadores, produções ou atuação profissional, conduzindo de certo modo, a características vistas no contexto de redes sociais.

Para realização deste estudo, considera-se na composição do cenário apresentado o constante crescimento da Plataforma Lattes que

por sua vez oferece em seu site dados e estatísticas sobre pesquisas de CT&I no Brasil atualizadas frequentemente permitindo a extração de informações quantitativas e qualitativas.

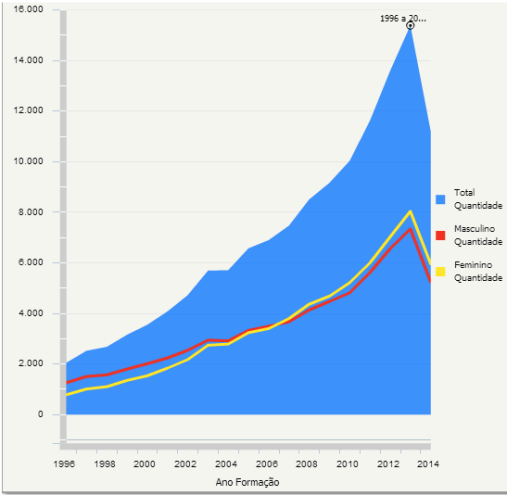
A avaliação do aumento quantitativo de informação na Plataforma Lattes pode ser realizada considerando os dados disponíveis no CNPq (CNPq, 2014a). As informações presentes na Figura 13 e Figura 14 referem-se ao segundo semestre de 2014 e consideram a formação de mestres e doutores.

Figura 13: Evolução quantitativa de cadastros no Lattes – Mestrado.



Fonte: CNPq (2014a).

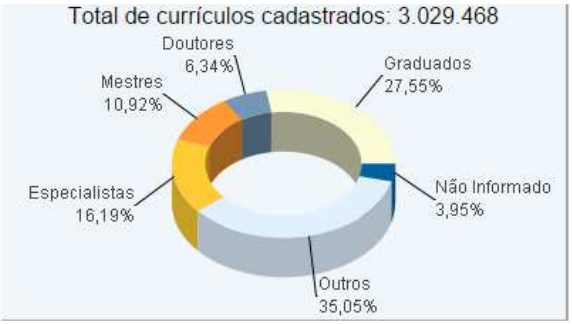
Figura 14: Evolução quantitativa de cadastros no Lattes – Doutorado.



Fonte: CNPq (2014a).

De acordo com os gráficos, entre 2010 e outubro de 2014 foram formados aproximadamente 12367 doutores e 35322 mestres no país por ano. De acordo com Silva (2012) em 2012 existiam 2.266.532 currículos cadastrados no Lattes, e hoje, de acordo com a Figura 15, este número apresentou um crescimento de 33,66%.

Figura 15: Quantidade de Currículos Cadastrados no Lattes.



Fonte: CNPq (2014a).

Considerando o aumento desta base de dados surge o desafio de prover informações gerenciais e que possibilitem análises mais

especializadas a partir da interrelação destes dados onde o modelo de banco de dados relacional passa a ser questionado e confrontado com modelos não relacionais (NoSQL).

Desta forma, através da definição de modelos comparativos entre Bancos de Dados Relacionais e NoSQL, mais especificamente orientados a grafo (Neo4j), este capítulo apresenta a avaliação do desempenho destes modelos considerando um contexto de alta demanda de informação com o objetivo de responder pergunta apresentada na problemática deste trabalho.

Para a avaliação deste trabalho foram baixados 100 currículos da base de dados Lattes manualmente. Para esta seleção foram pesquisados autores cujas publicações contivessem a palavra-chave "biotecnologia" através do Portal Inovação[1]. O conjunto resultante de pessoas foi anotado e pesquisado manualmente no portal do Lattes que por sua vez viabiliza o *download* dos arquivos XML.

Dos currículos considerados para a pesquisa foram extraídas as produções científicas e tecnológicas de cada CV-Lattes, informações relacionadas a frequência individual das palavras-chave por ano de publicação, bem como a frequência conjunta (coocorrência) entre duas palavras-chave quaisquer também por ano de publicação.

3.2 MODELO DE DADOS

Esta seção detalha os modelos de dados propostos ou adaptados neste trabalho para suporte ao estudo comparativo das abordagens relacional e orientada a grafo (não relacional).

3.2.1 Modelo de Dados Relacional

Como visto anteriormente, o modelo de dados relacional é composto por um conjunto de tabelas interconectadas representando determinado domínio de maneira estruturada.

O modelo de dados relacional considerado no trabalho e adaptado de Sérgio (2013) possui as seguintes informações:

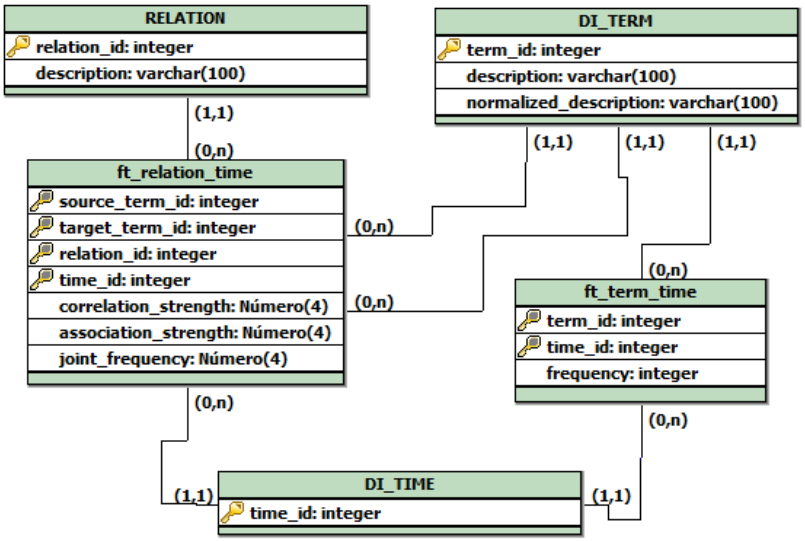
a) Cadastro de Termos: contém um identificador, a descrição natural e a descrição normalizada de determinada palavra-chave;

[1] http://www.portalinovacao.mct.gov.br

b) Cadastro de Período: estabelece o identificador do ano da publicação;
c) Relacionamento de Termos com Período: determina a frequência com que um termo ocorre num período específico;
d) Cadastro de Relacionamento: Determina o tipo do relacionamento existente entre os termos;
e) Cadastro de Relacionamento entre os Termos: contém a discriminação do termo de origem e destino, bem como o tipo de relacionamento utilizado, o período da ocorrência, a força de correlação, o peso de associação e frequência dos termos para o conjunto estabelecido.

De acordo com a proposição do modelo, o banco de dados relacional foi projetado para comportar as informações requeridas conforme especificado na
Figura 16:

Figura 16: Modelo lógico do Banco de Dados Relacional.



Fonte: Adaptado de Sérgio (2013).

O cadastro das palavras-chaves é realizado na tabela DI_TERM composta por um identificador sequencial (*term_id*) e dois campos

descritivos, sendo um a descrição do termo (*description)*, e o segundo a descrição normalizada (*normalized_description*), ou seja, sem acentuação e capitalizada. Esta tabela possui um índice para o campo descrição normalizada que serve de suporte às consultas realizadas para a recuperação dos termos pelo valor textual deste campo.

O cadastro de período é representado pela tabela DI_TIME identificada por um campo inteiro (*time_id*) que para o modelo proposto irá representar o ano da publicação científica e tecnológica.

O registro da ocorrência de um termo em um determinado período é representado pela tabela FT_TERM_TIME que possui dois campos como identificadores primários: (a) *term_id*, chave estrangeira da relação *di_term*; e (b) *time_id*, chave estrangeira da relação *di_time*. A tabela *dt_term_time* possui também um quantificador inteiro para indicação da frequência da ocorrência (*frequency*) de determinada palavra-chave por ano.

O registro de tipo de relacionamento existente entre os termos é representado pela tabela RELATION composta por um identificador sequencial (*relation_id*) e um campo descritivo (*description)*. Para o modelo proposto, será considerado apenas o relacionamento "estão relacionadas" indicando que duas palavras-chave quaisquer foram mencionadas em pelo menos uma produção.

A composição da coocorrência de dois termos em um determinado período é representada pela relação FT_RELATION_TIME. A tabela possui como identificadores: (a) termo de origem (*source_term_id*), chave estrangeira da relação *di_term*; (b) termo de destino (target_*term_id*), chave estrangeira da relação *di_term*; (c) o tipo do relacionamento (*relation_id*), chave estrangeira da relação *relation*; e (d) *time_id*, chave estrangeira da relação *di_time*. A tabela FT_RELATION_TIME conta ainda com três colunas quantitativas, sendo: (a) correlação da ocorrência conjunta dos termos de origem e destino (*correlation_strength*), não calculada neste trabalho; (b) associação da ocorrência conjunta dos termos de origem e destino (*association_strength*), não calculada neste trabalho; (a) frequência conjunta da ocorrência entre os termos de origem e destino (*joint_frequency*) calculada durante o processo de carga do modelo.

3.2.2 Modelo de Dados Orientado a Grafo

De acordo com a proposição do modelo utilizada para determinar o esquema de banco de dados relacional (Figura 16) foi também

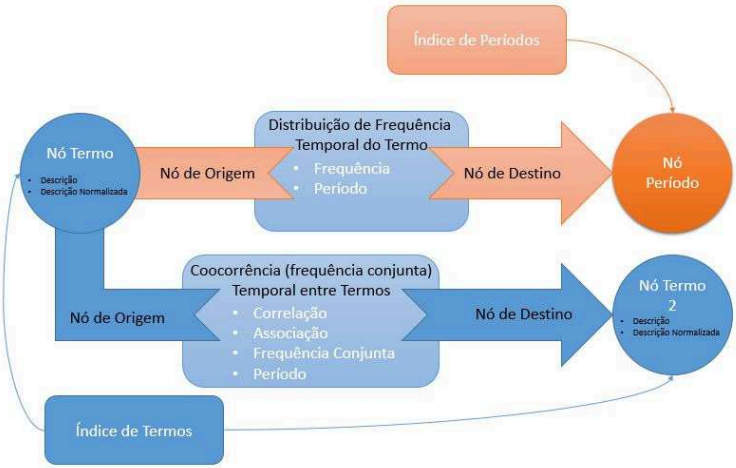
modelado o banco de dados orientado a grafo (Neo4j) onde os termos e as ocorrências deixam de estar embutidos em relações e se tornam objetos mapeados por índices (Figura 17).

Inicialmente foram criados dois índices, um para o mapeamento dos termos e outro para mapeamento dos períodos, desta forma, todo termo/período inserido passa a ser persistido no índice para posterior mapeamento/consulta.

No modelo proposto os índices são representados por barras horizontais que apontam para os nós de acordo com suas respectivas "classes". O índice representado em azul mapeia os termos inseridos, enquanto o índice representado laranja mapeia o *nó* com a informação do período.

Os termos são indicados por *nós* azuis, o objeto indicativo de período em um nó alaranjado, sendo que por conta da diferença da distribuição e organização dos dados deixa de existir uma tabela específica para o cadastro de termos e a distinção dos nós passa a ser realizada através do índice ao qual o nó está vinculado, bem como ao conjunto de propriedades que o nó possui. Independente do conteúdo, cada *nó* recebe um identificador único do SGBD Neo4j válido para todo o banco de dados.

Figura 17: Modelo proposto no formato de grafo.



Fonte: Autor.

Os nós do tipo termo possuem as seguintes informações: (a) identificador interno; (b) descrição (*description*); (c) descrição normalizada (*normalized_description*).

Os nós do tipo período, representados em laranja, possuem apenas um identificador interno e um campo indicativo da unidade de tempo utilizada, neste caso o ano.

O grande diferencial entre os dois modelos se dá no relacionamento entre os nós. Em azul está representado o relacionamento de coocorrência temporal entre termos e em laranja a distribuição de frequência temporal do termo em si.

Como nos bancos de dados orientados a grafo o relacionamento não representa apenas um apontamento, mas um objeto, este pode conter dados a respeito da relação estabelecida.

No caso do relacionamento de coocorrência, os dados armazenados na relação são: (a) identificador único da relação para o SGDB Neo4j; (b) o tipo do relacionamento; (c) a correlação da ocorrência, não calculada neste trabalho; (d) a associação da ocorrência, não calculada neste trabalho; (e) a frequência conjunta da ocorrência, calculada durante o processo de carga do XML; e (f) o período (ano) em que a coocorência foi efetuada. Diferentemente do modelo relacional, os termos de origem e destino (*source/target*) não são representados como colunas estrangeiras, mas como componentes internos do relacionamento que indicam quais são os nós a ele vinculados.

No relacionamento de distribuição de frequência temporal do termo além do ponteiro para o nó de origem (*source*) e do ponteiro para o nó de destino (*target*), existem dois campos de dados, o tempo da ocorrência (*year*), representado por um valor inteiro, e a frequência desta ocorrência no período (*frequency*).

A partir dos modelos relacional e orientado a grafo propostos, foi desenvolvido um protótipo para realização das simulações de carga de dados necessárias para a avaliação deste trabalho.

3.3 DETALHAMENTO DO PROTÓTIPO

Para o desenvolvimento do protótipo foi utilizada a linguagem de programação Java uma vez que este suporta tanto o estabelecimento de uma conexão JDBC (*Java Database Connectivity)* com o banco de dados relacional, quanto a conectividade com o banco de dados Neo4j® utilizando-se do método *embedded* que instancia o banco de dados pelo

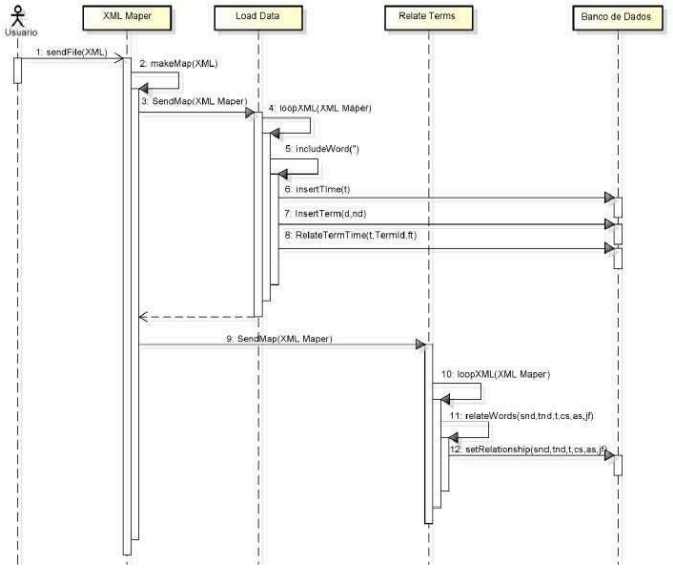
próprio Java® de forma encapsulada. O banco de dados Oracle XE® versão 10 foi utilizado para suportar o estudo do modelo relacional.

Considerando que a fonte dos dados utilizada neste trabalho é composta por uma série de arquivos XMLs foi utilizado um método para composição destes em uma *hash*, a partir da qual foram alimentados os bancos de dados relacional e orientado a grafo utilizando o mesmo tipo de comparação em ambos os ambientes. De modo geral, os dados semiestruturados em XML são integrados em memória através de um conjunto de objetos e estruturas do tipo *hash* facilitando a manipulação das informações que serão persistidas nos dois modelos de banco de dados.

A população dos bancos de dados relacional e orientado a grafo seguiram a mesma metodologia de forma que se possa comparar os resultados em cenários semelhantes, e representar o comportamento através de diagramas de atividade e sequência genéricos.

Para a população do banco de dados é realizada a chamada de um método para incluir a listagem de termos relacionados (Figura 18), bem como, o período de sua ocorrência e consequentemente a composição (distribuição temporal) das frequências de cada um dos termos.

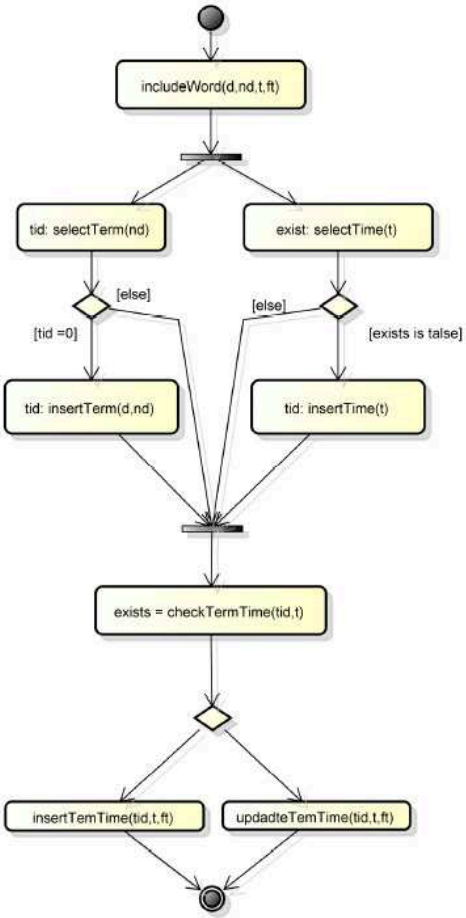
Figura 18: Diagrama de Sequência.



Fonte: Autor.

A partir da Figura 18 foi gerado um diagrama de atividades para representar persistência dos termos (Figura 19) e outro para representar o relacionamento entre os termos (Figura 20).

Figura 19: Diagrama de Atividades – Persistência de Termos.



Fonte: Autor.

No diagrama representado pela Figura 19, quando a função de inclusão de termos é chamada, ela realiza a instanciação das classes de persistência de termo, período e de composição de distribuição de

frequência de determinado termo pelo ano. Uma vez criados os mecanismos de persistência é realizada a verificação da existência do período em questão e, caso o período não exista, ele é inserido. De maneira similar ao período, é verificada a existência do termo na base de dados, se não existente este será inserido e seu identificador primário será recuperado.

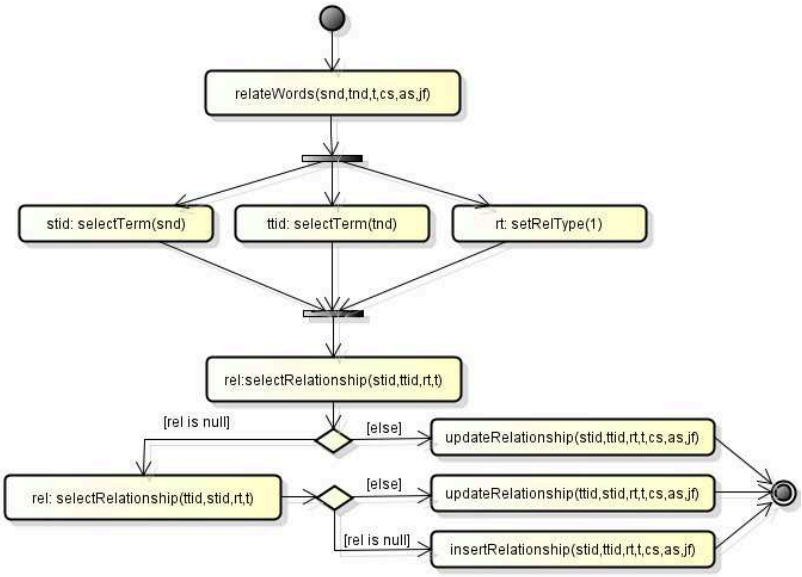
Após a persistência do termo e do período é realizada a verificação na base de dados da existência do relacionamento de ocorrência para o referido termo no determinado período. Se a relacionamento é inexistente este é inserida, caso contrário, é incrementado somando ao valor atual o novo valor computado. Vale mencionar que o valor computado, por exemplo, o total de vezes que determinado termo ocorre em um determinado ano para um currículo em particular, já se encontra agregado na memória como resultado do processo inicial de leitura do arquivo XML.

De acordo com o diagrama apresentado na Figura 18, após a persistência de todos os termos, o XML a tabela *hash* são novamente percorridos para a composição da coocorrência dos termos em um determinado momento.

Para criação do relacionamento de coocorrência entre os termos (Figura 20) os identificadores dos termos de origem (*source*) e destino (*target*) são recuperados do banco de dados, o tipo de relacionamento recebe um código identificando um valor padrão (um relacionamento identificado como geral, ou seja, indica simplesmente que existe coocorrência entre termos em um mesmo item de produção C&T) e é realizada a verificação de existência do relacionamento entre os termos.

Caso a busca de relacionamento entre os termos origem e destino seja inexistente, é submetida uma nova pesquisa invertendo a definição do termo de origem pelo termo de destino. Esta operação é necessária uma vez que, no modelo relacional, determinado termo pode aparecer tanto no campo origem quanto destino. Adicionalmente, para o cenário deste trabalho um relacionamento é sempre tratado como algo geral, ou seja, sem rótulo específico e com a frequência conjunta (coocorrência) independente da direção. Se em alguma das consultas houver retorno de dados a frequência conjunta do relacionamento é atualizada, senão o relacionamento é criado.

Figura 20: Diagrama de Atividades – Relacionamento de Termos.



Fonte: Autor

Em ambos os modelos o controle transacional foi realizado de forma que, para cada inclusão de termos, período, relacionamento da distribuição de frequência entre termo e período, bem como a coocorrência entre os termos de forma cronológica ocorreu isoladamente com persistência dos dados a cada operação.

Conforme ilustrado anteriormente, ao estabelecer os relacionamentos se fez necessária a recuperação do(s) termo(s) pela descrição normalizada. A recuperação destes termos foi suportada por um índice tanto no modelo relacional quanto no modelo orientado a grafo.

3.4 ANÁLISE DOS RESULTADOS

Para avaliação do comportamento de carga nos modelos acima apresentados para o cenário proposto foram submetidos 100 currículos aos bancos de dados em três cargas distintas (10, 50 e 100 currículos) sendo que cada carga foi individualmente mensuradas.

Como cada um dos currículos possuem uma quantidade diferenciada de termos relacionados, a avaliação da capacidade de carga gradual dos currículos não pode ser medida simplesmente pela quantidade de currículos submetidos. Portanto, para a submissão dos currículos foi montado um vetor ordenado para a execução das três cargas: a) do primeiro ao décimo currículo; b) do primeiro ao quinquagésimo currículo; e c) do primeiro ao centésimo currículo.

Para análise de resultados a coleta de dados foi dividida em três etapas para cada uma das execuções: a) tempo dispendido para inserir os termos; b) tempo dispendido para relacionar a distribuição de frequência dos termos de forma temporal; c) o tempo dispendido para relacionamento da coocorrência temporal entre os termos; e d) o tempo total de execução.

Em cada um dos períodos medidos foram coletados os dados referentes ao ápice de utilização de CPU, disco e memória de forma percentual para as fases de inserção de termos, relacionamento de termos aos períodos e o relacionamento temporal entre os termos (coocorrência temporal). A média dos tempos das três etapas foi também informada em cada uma das simulações. As aferições do consumo de CPU, disco e memória foram realizadas tanto para o modelo relacional e quanto para o modelo não relacional (orientado grafo)

Na carga realizada para o modelo relacional representado na Tabela 9 pode-se observar que a etapa em que ocorre a maior utilização de recursos computacionais (CPU, disco e memória) é a etapa de relacionamento da coocorrência temporal entre os termos. Ainda nesta etapa, apesar de uma distribuição equilibrada do uso do processador há um aumento no uso da memória, bem como, pode-se observar que há um agravante no momento em que é realizada a persistência em disco que reflete negativamente no tempo de execução total.

Tabela 9: Medição da população do banco de dados relacional.

Relacional						
Currículos	**Tempo (hh:mm:ss)**		**Quantidade de registros**	**CPU**	**Disco**	**Memória**
1 a 10	**Inserir Termos**	00:00:04	1813(t)	8%	42%	32%
	Relacionar Termos e Período	00:00:06	38 (p) 12853 (rtp)	12%	47%	44%
	Relacionar Termos	00:00:29	5987 (rtt)	12%	44%	43%
	Tempo Total	**00:00:39**	**Media**	**11%**	**44%**	**40%**
1 a 50	**Inserir Termos**	00:00:11	8313 (t)	19%	54%	45%
	Relacionar Termos e Período	00:00:30	44 (p) 87715 (rtp)	18%	48%	56%
	Relacionar Termos	00:04:04	29988 (rtt)	18%	100%	56%
	Tempo Total	**00:04:45**	**Media**	**18%**	**67%**	**52%**
1 a 100	**Inserir Termos**	00:00:23	13915 (t)	21%	58%	58%
	Relacionar Termos e Período	00:00:52	48 (p) 163824 (rtp)	18%	99%	58%
	Relacionar Termos	00:06:47	52280 (rtt)	17%	100%	58%
	Tempo Total	**00:07:42**	**Media**	**19%**	**86%**	**58%**

Fonte: Autor.

Legenda: (t) representa a unidade de medida que representa a quantidade de termos; (p) a unidade de medida quantitativa de períodos (quantidade de anos); (rtp) a unidade de medida quantitativa de relacionamento de frequência do termo; e (rtt) a unidade de medida quantitativa da coocorrência temporal entre termos.

Não diferentemente do modelo relacional para o modelo de banco de dados orientado a grafo a etapa de persistência dos dados em disco se revela como um fator de queda de desempenho. Pela diferente forma de gerenciamento dos dados realizada pelo Neo4j®, o teste realizado demonstra que para o cenário proposto o comportamento deste se revela como insuficiente. No Neo4j a taxa de utilização de disco supera a medição da carga do modelo relacional quando considerado 100

currículos, apesar de possuir um consumo menor dos recursos de memória e CPU (Tabela 10).

Tabela 10: Medição da alimentação do banco de dados orientado a grafo

Grafo						
Currículos	**Tempo (hh:mm:ss)**		**Quantidade de registros**	**CPU**	**Disco**	**Memória**
1 a 10	**Inserir Termos**	00:03:58	1815 (n) 34353 (p)	**3%**	**87%**	**47%**
	Relacionar Termos e Período	00:03:21	18840 (rts)	3%	100%	48%
	Relacionar Termos	00:05:47		5%	100%	52%
	Tempo Total	**00:13:06**	**Media**	**3,7%**	**95,6%**	**49%**
1 a 50	**Inserir Termos**	00:16:03	8315 (n) 34353 (p)	3%	99%	47%
	Relacionar Termos e Período	00:17:42	117703 (rts)	3%	100%	50%
	Relacionar Termos	00:53:47		4%	100	53%
	Tempo Total	**01:27:32**	**Media**	**3,33%**	**99,67%**	**50%**
1 a 100	**Inserir Termos**	00:31:56	13917 (n) 401360 (p)	4%	99%	49%
	Relacionar Termos e Período	00:32:38	216104 (rts)	4%	99%	51%
	Relacionar Termos	01:33:08		6%	100%	55%
	Tempo Total	**02:37:42**	**Media**	**4,6%**	**99,33%**	**51,67%**

Fonte: Autor

Legenda: (n) unidade de medida quantitativa de nós (nodo); (p) unidade de medida quantitativa de períodos; (rts) unidade de medida quantitativa de relacionamentos de frequência do termo e de coocorrência temporal entre termos.

A diferença de desempenho em tempo de execução entre os dois modelos para cada uma das cargas foi mensurada conforme a Tabela 11.

Tabela 11: Comparativo entre as medições dos modelos relacional e orientado a grafo

Comparativo				
Currículos	**Etapa**	**Tempo Relacional**	**Tempo Grafo**	**Diferença**
1 a 10	**Inserir Termos**	00:00:04	00:03:58	5850%
	Relacionar Termos e Período	00:00:06	00:03:21	3250%
	Relacionar Termos	00:00:29	00:05:47	1097%
	Tempo Total	00:00:39	00:13:06	1915%
1 a 50	**Inserir Termos**	00:00:11	00:16:03	8655%
	Relacionar Termos e Período	00:00:30	00:17:42	3440%
	Relacionar Termos	00:04:04	00:53:47	1223%
	Tempo Total	00:04:45	01:27:32	1743%
1 a 100	**Inserir Termos**	00:00:23	00:31:56	8230%
	Relacionar Termos e Período	00:00:52	00:32:38	3665%
	Relacionar Termos	00:06:47	01:33:08	1273%
	Tempo Total	00:07:42	02:37:42	1948%

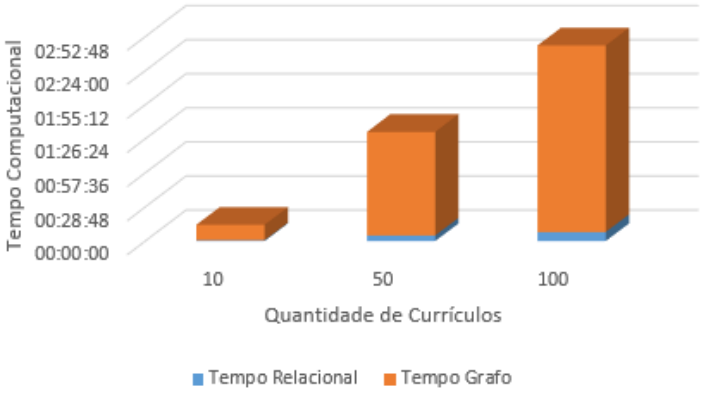
Fonte: Autor

Através da comparação de tempo realizada podemos observar que a diferença para inserir termos, que gera acesso direto a disco, teve uma taxa percentual de 5850% a 8655% entre os modelos.

De acordo com a ilustração realizada na Figura 21, onde o tempo despendido pelo modelo orientado a grafo está representado em alaranjado e o tempo despendido pelo modelo relacional em azul, pode-

se observar que o tempo total da inclusão dos dez primeiros currículos no banco de dados orientado a grafo foi superior ao tempo total da inclusão de todos os currículos no modelo relacional.

Figura 21: Gráfico comparativo entre as medições dos modelos relacional e orientado a grafo.



Fonte: Autor

Por conta da grande diferença apresentada entre os modelos realizando o controle de transação e recuperando os índices do Neo4j a cada transação foram realizadas duas novas otimizações no modelo de dados orientado a grafo a fim de investigar o porquê de tamanha diferença entre os modelos. A Tabela 12 demonstra a mesma medição realizada anteriormente para o modelo de dados orientado a grafo, contudo, agora o índice de mapeamento de períodos e termos é mantido em memória não se fazendo necessário a cada transação restaurar o índice para realizar as consultas desejadas. Como pode-se observar, esta estratégia apesar de reduzir o tempo de relacionamento, distribuição de frequência dos termos e da otimização do uso da capacidade de processamento, não surtiu resultados esperados quando observado o tempo total de processamento.

Tabela 12: Medição da alimentação do banco de dados orientado a grafo mantendo o índice em memória

Grafo - Índice em memória						
Currículos	**Tempo (hh:mm:ss)**		**Quantidade de registros**	**CPU**	**Disco**	**Memória**
1 a 10	**Inserir Termos**	00:04:13	1815 (n) 34353 (p)	3%	73%	47%
	Relacionar Termos e Período	00:04:08	18840 (rts)	5%	99%	48%
	Relacionar Termos	00:07:52		7%	99%	60%
	Tempo Total	**00:16:13**	**Media**	**5%**	**99%**	**51,67%**
1 a 50	**Inserir Termos**	00:22:22	8315 (n) 34353 (p)	4%	99%	52%
	Relacionar Termos e Período	00:17:02	117703 (rts)	6%	99%	54%
	Relacionar Termos	00:50:06		8%	100%	62%
	Tempo Total	**01:29:30**	**Media**	**6%**	**99,33%**	**56%**
1 a 100	**Inserir Termos**	00:35:16	13917 (n) 401360 (p)	4%	99%	53%
	Relacionar Termos e Período	00:31:37	216104 (rts)	8%	100%	53%
	Relacionar Termos	01:46:54		10%	100%	65%
	Tempo Total	**02:53:47**	**Media**	**7,33%**	**99,33%**	**57%**

Fonte: Autor

Como última análise comparativa entre os modelos e a partir da constatação da ineficiência do banco de dados orientado a grafo (Neo4J®) no controle de múltiplas persistências operacionais contra o banco de dados relacional, foi realizada uma segunda otimização nos métodos de persistência de dados. Nesta segunda otimização a persistência em disco foi realizada em apenas três momentos, ou seja, ao

final de cada uma das etapas (Inserir Termos, Relacionar Termos e Período e Relacionar Termos) apresentadas na Tabela 13.

Tabela 13: Medição da alimentação do banco de dados orientado a grafo com persistência postergada por bloco de operações

Grafo - Índice em memória e persistência postergada						
Currículos	**Tempo (hh:mm:ss)**		**Quantidade de registros**	**CPU**	**Disco**	**Memória**
1 a 10	**Inserir Termos**	00:00:08	1815 (n) 34353 (p)	21%	65%	40%
	Relacionar Termos e Período	00:00:02	18840 (rts)	25%	39%%	41%
	Relacionar Termos	00:00:05		39%	42%	56%
	Tempo Total	**00:00:15**	**Media**	**28,33%**	**48,67%**	**45,67%**
1 a 50	**Inserir Termos**	00:00:18	8315 (n) 34353 (p)	32%	53%	47%
	Relacionar Termos e Período	00:00:06	117703 (rts)	68%	34%	48%
	Relacionar Termos	00:00:21		41%	51%	69%
	Tempo Total	**00:00:45**	**Media**	**47%**	**46%**	**54,67**
1 a 100	**Inserir Termos**	00:00:16	13917 (n) 401360 (p)	36%	58%	51%
	Relacionar Termos e Período	00:00:06	216104 (rts)	69%	42%	55%
	Relacionar Termos	00:00:49		49%	60%	73%
	Tempo Total	**00:01:11**	**Media**	**51,33%**	**53,33%**	**59,67%**

Fonte: Autor

Com base nos resultados da comparação apresentada na Tabela 14 pode-se constatar que há de fato um grande ganho de desempenho desde a inserção até os relacionamentos de distribuição de frequência e coocorrência dos termos no modelo de dados orientado a grafo. Isto ocorre principalmente pelo fato de que os múltiplos relacionamentos são tratados em memória. Contudo, pode-se verificar quando comparado ao primeiro caso que existe uma defasagem do modelo orientado a grafo

quando ocorre a necessidade de persistir massivamente os dados em disco. Mesmo neste cenário otimizado o modelo orientado a grafo, quando se analisam as etapas de inserção de termos possui um incremento de 100% e 64% quanto considerados 10 e 50 currículos, respectivamente. Essa tendência se inverte quando considerado 100 currículos, em que o modelo em grafo apresenta uma redução de 30% do tempo.

Tabela 14: Comparativo entre as medições dos modelos relacional e orientado a grafo otimizado (persistência postergada).

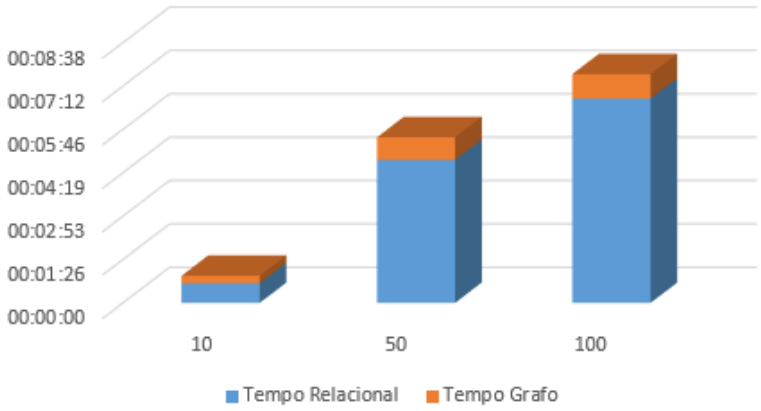
Comparativo com Grafo Otimizado				
Currículos	**Etapa**	**Tempo Relacional**	**Tempo Grafo**	**Diferença**
1 a 10	**Inserir Termos**	00:00:04	00:00:08	100%
	Relacionar Termos e Período	00:00:06	00:00:02	-67%
	Relacionar Termos	00:00:29	00:00:05	-83%
	Tempo Total	00:00:39	00:00:15	-62%
1 a 50	**Inserir Termos**	00:00:11	00:00:18	64%
	Relacionar Termos e Período	00:00:30	00:00:06	-80%
	Relacionar Termos	00:04:04	00:00:21	-91%
	Tempo Total	00:04:45	00:00:45	-84%
1 a 100	**Inserir Termos**	00:00:23	00:00:16	-30%
	Relacionar Termos e Período	00:00:52	00:00:06	-88%
	Relacionar Termos	00:06:47	00:00:49	-88%
	Tempo Total	00:07:42	00:00:49	-89%

Fonte: Autor.

O ganho, portanto, pode ser devidamente observado nas etapas de relacionamento dos termos tendo produzir uma redução no tempo em torno de 90%. Nota-se a partir do cenário uma tendência na redução do tempo à medida que o aumento da massa de dados ocorre.

De forma geral, quando analisado este último cenário, ilustrado na Figura 22, comparando-se os dois modelos, observa-se uma redução no tempo total para o modelo orientado a grafos para cada uma das etapas de 62%, 84% e 89%, respectivamente, de forma que a representação demonstrada na Figura 22 quando comparada ao resultado apresentado na Figura 21 tem-se uma inversão da ocupação temporal do grafo entre os modelos relacional e orientado a grafo.

Figura 22: Gráfico comparativo entre as medições dos modelos relacional e orientado a grafo otimizado (persistência postergada).



Fonte: Autor.

4 CONSIDERAÇÕES FINAIS

O objetivo geral deste trabalho foi realizar um estudo comparativo entre as abordagens de Banco de Dados Relacional e NoSQL considerando aspectos gerais de desempenho no que tange a população de dados.

A partir do levantamento bibliográfico nas áreas de Banco de Dados com foco principal em Banco de Dados NoSQL buscou-se entender o domínio e o contexto do trabalho de forma que a partir deste conhecimento fosse possível implementar um protótipo capaz de mensurar dados quantitativos para atender o objetivo proposto.

O protótipo proposto foi aplicado em um cenário real sobre as bases de dados Lattes do CNPq para realizar o mapeamento de frequência individual de termos e frequência conjunta (coocorrência) entre termos. Para a avaliação de desempenho foram realizados estudos com 10, 50 e 100 currículos cujo modelo de dados relacional foi baseado no trabalho desenvolvido por Sérgio (2013) e a partir deste, foi também modelada uma estrutura de dados orientada a grafo (NoSQL).

Considerando a implementação do protótipo foi possível analisar a coleta de resultados estabelecendo parâmetros comparativos sobre o desempenho apresentado nas duas estruturas/modelos de dados estudados.

O modelo de dados relacional apresentou um desempenho superior em comparação ao modelo de dados orientado a grafo (Neo4j) quando a implementação da carga de dados estabelecia o controle de persistência termo a termo. Apesar disto, percebe-se que com o aumento da carga o processo se tornava proporcionalmente mais lento no modelo orientado a grafo.

O modelo de dados orientado a grafo mostrou-se muito eficiente somente quando o controle de persistência de dados foi realizado em grandes grupos de dados, diminuindo o acesso a disco e mantendo os dados em memória de forma que quanto maior o volume de dados submetido maior a vantagem percentual do grafo sobre o banco de dados relacional.

Cabe salientar que apesar da aparente vantagem dos bancos de dados orientados a grafo para o segundo cenário isto exigiria, para uma base com milhões de currículos, uma capacidade considerável para armazenamento de dados em memória. Levando-se em conta o atual estágio atual tecnológico, tal cenário ainda constitui-se em um desafio computacional.

Apesar do atual desafio de se obter uma alta capacidade de armazenamento de dados em memória, o que pode comprometer também a durabilidade dos dados que é um dos pilares das características ACID, pode-se vislumbrar um futuro promissor tendo em vista as recentes descobertas/propostas de integração das tecnologias SSD à memória RAM. Exemplo dessa tecnologia vem sendo desenvolvida pela empresa Apacer® capaz de prover uma capacidade superior a 512 GB de memória com tendência de em breve alcançar o horizonte de 1 TB (CUNNINGHAM, 2014). Considerando estruturas distribuídas de alto desempenho, a capacidade de armazenar dados em memória com capacidade de persistência tende a crescer nos próximos anos.

De modo geral, pode-se concluir que o modelo orientado a grafo apresenta desempenho superior para lidar com dados interconectados quando considerando a possibilidade de manter as informações em memória. No entanto, considerando as tecnologias computacionais disponíveis no mercado e com base nos resultados do estudo realizado pode-se considerar que o modelo de dados relacional ainda é uma estrutura extremamente eficiente e capaz de comportar problemas complexo. Por fim, vale mencionar que o avanço do conceito de bancos de dados em memória irá impactar positivamente em modelos relacionais e não relacionais.

Durante o desenvolvimento deste trabalho e pensando em perspectivas de pesquisa foram vislumbrados alguns trabalhos futuros. Entre as possibilidades encontra-se a continuidade deste trabalho no que tange ao aumento no número de Currículos Lattes para alguns milhares. Para tal, seria necessária uma estrutura capaz de manter os dados em memória.

Outra possibilidade reside na expansão da análise realizada neste trabalho comparando a abordagem relacional com outros modelos NoSQL. Inclui-se a isto a necessidade de avaliação de abordagens recentes classificadas como NewSQL que mesclam alta disponibilidade e escalabilidade com os requisitos de transacionais e de consistência dos dados.

Por fim, estudos em diferentes domínios utilizando a abordagem orientada a grafo podem ser realizados. Entre estes domínios encontram-se a aplicação desse modelo em problemas que envolvam os conceitos de Web 2.0, tais como, os Sistemas de Recomendação. Mesmo em cenários de tomada de decisão mais clássicos, como análise e simulação de impactos econômicos, podem ser modelados como grafos.

REFERÊNCIAS

ABADI, D. J.. Data Management in the Cloud: Limitations and Opportunities. Data Engineering Bulletin, v.32, n.1, p. 3-12, 2009.

ABREU, E. S.. Armazenamento e processamento de grandes grafos em bancos de dados geográficos. 2013. 113 f. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais - INPE. São José dos Campos, SP, Brasil. 2013.

ANTON, H.; RORRES, C. Álgebra Linear: com aplicações. 10. ed. Porto Alegre: Bookman, 2012. 768p.

ASSUNÇÃO, M. D.; CALHEIROS, R. N.; BIANCHI, S; NETTO, M.A.S.; BUYYA, R.. Big Data Computing and Clouds: Challenges, Solutions, and Future Directions. Journal of Parallel and Distributed Computing, In Press, 2014.

ÁVILA, M. G.; GROWNWALD, C. L.O.. História da matemática e resolução de problemas: Uma aliança possível 2004. 185f. Dissertação (Pós - Graduação em Ensino de Ciências e Matemática) - Universidade Luterana do Brasil, Canoas. 2004.

BARROS, E. M.; Cloud computing: Tendência de sucesso nas operações de outsourcing em TI. 2010. Disponível em: < http://convergecom.com.br/tiinside/19/07/2010/cloud-computing-tendencia-de-sucesso-nas-operacoes-de-outsourcing-em-ti/#.VHMk44vF9tA>. Acesso em: Novembro de 2014.

BIANCO, Rafael. Disponibilização de Dados Abertos Utilizando Linked Data: Uma avaliação Teórico-Prática. 2011. 175f. Trabalho de Conclusão de Curso - UFSC, Florianópolis, 2011.

CAMPBELL-KELLY, M., The RDBMS Industry: A Northern California Perspective. IEEE Annals of the History of Computing, v. 34, n. 4, p. 18-29, 2012.

CARDOSO, D. M.. Teoria dos Grafos e Aplicações. 2005. 99 f. Dissertação (Mestrado em Matemática) - Departamento de Matemática, Universidade de Aveiro. Aveiro, Portugal. 2005.

CAVALCANTE, F. N. S.; SILVA, S. D. Grafos e suas aplicações. 2009. 64f. Trabalho de Conclusão de Curso. Centro Universitário Adventista de São Paulo. São Paulo. 2009.

CNPQ. Evolução da Formação de Mestres e Doutores no Brasil. 2014a. Disponível em: <http://estatico.cnpq.br/painelLattes/evolucaoformacao/>. Acesso em: 18 de outubro 2014

CNPQ. Sobre a Plataforma Lattes. 2014b. Disponível em: <http://www.cnpq.br/web/portal-lattes/sobre-a-plataforma>. Acesso em: 11 de outubro de 2014.

CODD, E. F., A Relational Model of Data for Large Shared Data Banks, Comm. ACM, v. 13, p. 377–387, 1970.

COSTA, E. R.S. Uma proposta de ensino de análise combinatória para alunos do ensino médio. 2013. 108f. Trabalho de Conclusão de Curso - UFLA, Lavras, 2013.

CUNNINGHAM, A. Mount your hard drive... on your RAM? New "SDIMM" sticks include M.2 SSD slots to save room inside your case. Disponível em: <http://arstechnica.com/gadgets/2014/08/mount-your-hard-drive-on-your-ram-with-apacers-new-memory-sticks/>. Acesso em: Novembro de 2014.

DARWEN, H., The Relational Model: Beginning of an Era. IEEE Annals of the History of Computing, v. 34, n. 4, p. 7-8, 2012.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistema de Banco de Dados. Revisor técnico Luíz Ricardo de Figueiredo. São Paulo: Pearson Addison Wesley. 2005.

GERSTING, Judith L. Fundamentos matemáticos para a ciência da computação. 3ª Edição, Rio de Janeiro: Livros Técnicos e Científicos, Editora S.A., 2001.

GIL, A. C. Como elaborar projetos de pesquisa. 4th ed. São Paulo: Atlas, 2008.

GRAD, B., Relational Database Management Systems: The Formative Years. IEEE Annals of the History of Computing, v. 34, n. 4, p. 7-8, 2012.

GRIER, D. A. The Relational Database and the Concept of the Information System. IEEE Annals of the History of Computing, v. 34, n. 4, p. 9-16, 2012.

HADJIGEORGIOU, C.. RDBMS vs NoSQL: Performance and Scaling Comparison, 2013, 61f. MSc in High Performance Computing, The University of Edinburgh, United Kingdom.

HAN, J.; HAIHONG, E.; GUAN, L.; DU, J.; Survey on NoSQL database. 6th International Conference on Pervasive Computing and Applications (ICPCA), p. 363-366, IEEE, 2011.

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N.B. MOKHTAR, S.; GANI, A.; A, KHAN, S. U. The rise of “big data” on cloud computing: Review and open research issues, Information Systems, v. 47, p. 98-115, 2015.

HECHT, R., JABLONSKI, S; NoSQL evaluation: A use case oriented survey. International Conference on Cloud and Service Computing (CSC), p. 336-341, IEEE, 2011.

HERNANDES, F.; BERTON, L.; CASTANHO, M. J. P.. O problema de caminho mínimo com incertezas e restrições de tempo. Pesquisa Operacional, Rio de Janeiro, v.29, n.2, 2009.

HEUSER, C. A.. Projeto de Banco de Dados. 6 ed. Porto Alegre: Bookman, 2009.

IBM. What Is Big Data: Bring Big Data to the Enterprise, 2014, Disponível em: <http://www-01.ibm.com/software/au/data/bigdata/>, Acessado: 22 de janeiro de 2014.

KATZ, D. S.; JHA, S.; PARASHAR, M.; RANA, O.; WEISSMAN, J. Survey and Analysis of Production Distributed Computing Infrastructures. Disponível em: http://arxiv.org/abs/1208.2649v1, Acesso em: Setembro de 2013.

KRISHNA, P. R.; VARMA, K. I.. Cloud analytics: A path towards next generation affordable BI, White paper, Infosys (2012).

KUROSE, J. F.; ROSS, K. W. Redes de Computadores e a Internet: uma nova abordagem. Tradução de Arlete Simille Marques. São Paulo: Wagner Luiz Zucchi, 2009. cap. 4, p. 228 - 305.

LAI, E. Computerworld: Twitter growth prompts switch from MySQL to 'NoSQL' database. 2010. Disponível em: http://www.computerworld.com/s/article/9161078/Twitter_growth_prompts_switch_from_MySQL_to_NoSQL_database. Acesso em: Março de 2014.

LAKSHMAN, A.; MALIK, P. Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, v. 44, n. 2, 2010.

LYMAN, Peter; VARIAN, Hal R. How much information? Executive summary. 2003.

NEO4J, The Neo4j Manual. Disponível em: <http://neo4j.com/docs/stable/>, Acesso em: Julho de 2014.

NEUBAUER, Peter. Graph Databases, NOSQL and Neo4j, 2010. Disponível em: <http://www.infoq.com/articles/graph-nosql-neo4j>. Acesso em: Outubro de 2014.

OLIVEIRA, D. BIG DATA: O desafio de garimpar informações. Computer World, n. 554, p. 12-17, 2013.

PREGER, Robert, The Oracle Story, Part 1:1977-1986. IEEE Annals of the History of Computing, v. 34, n. 4, p. 51-57, 2012.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. Sistemas de gerenciamento de banco de dados. 3. ed. São Paulo: McGraw-Hill Medical, 2008.

RAMOS, José Yoshiriro Asisaka; NASCIMENTO, Ana de Farias. NoSQL: Conceitos e Evolução. MUNDOJ - NoSQL: um novo paradigma ara persistência distribuída e escalável. Rio de Janeiro, n. 51, p. 6-9, jan./fev. 2012.

ROBINSON, J; WEBBER, J.; EIFREM, E. Graph Databases. 1ª Edição, Sebastopol, CA: O'Reilly, 2013.

ROWE, L. A. Relational Database Management Systems: The Formative Years. IEEE Annals of the History of Computing, v. 34, n. 4, p. 7-8,2012.

RUSSOM, P., Analytics: A New Nexus for Business Intelligence, TDWI best practices report, The Data Warehousing Institute (TDWI) Research, v. 8, n. 1, p. 9-12, 2011.

SAKR, S., LIU, A. ; BATISTA, D.M. ; ALOMARI, M.. A Survey of Large Scale Data Management Approaches in Cloud Environments. Communications Surveys & Tutorials, v. 13, n. 3, IEEE, 2011.

SCHOMM, F.; STAHL, F.; VOSSEN, G. Marketplaces for data: An initial survey, SIFMOD Record, v. 42, n. 1, p. 15-26, 2013.

SÉRGIO, M. C. Uma Arquitetura de Descoberta de Conhecimento baseada na Correlação e Associação Temporal de Padrões Textuais. 2013. 125f. Trabalho de Conclusão de Curso - UFSC, Araranguá, 2013.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. Sistema de Banco de Dados. 5. ed. Rio de Janeiro: Elsevier, 2006.

SILVA, E. L. da; MENEZES, E. M. Metodologia da pesquisa e elaboração de dissertação. 4. ed. rev. atual. Florianópolis: UFSC, 2005. 138 p.

SILVA, L. Uma Ontologia para Representação do Currículo Lattes. 2013. 120f. Trabalho de Conclusão de Curso - UFSC, Araranguá, 2013.

STROZZI, Carlo. NoSQL: a non-SQL RDBMS.1998. Disponível em: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page. Acesso em: Janeiro de 2014.

SUMATHI, S.; ESAKKIRAJAN, S. Fundamental of Relational Database Management Systems. Studies in Computational Intelligence, v. 47, Springer, 2007.

VICKNAIR, C.; MACIAS, M.; ZHAO, Z.; NAN, X.; CHEN, Y.; WILKINS, D.. A comparison of a graph database and a relational

database: a data provenance perspective. Proceedings of the 48th Annual Southeast Regional Conference (ACM SE ’10), n. 42, 2010.

WADE, B. W., Compiling SQL into System/370 Machine Language. IEEE Annals of the History of Computing, v. 34, n. 4, p. 49-50, 2012.

WADE, B.; CHAMBERLIN, D.; IBM Relational Database Systems: The Early Years. IEEE Annals of the History of Computing, v. 34, n. 4, p. 38-48, 2012.

WITTEN, I. H., FRANK, E., HALL, M. A., Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann, 2011, 664p.

WU, X.; ZHU, X.; WU, G. Q.; DING, W. Data Mining with Big Data. IEEE Transactions on Knowledge and Data Engineering, v. 26, n. 1, p. 97-207, 2014.

ZIKOPOULOS, P.; EATON, C.; ROOS, D.; DEUTSCH, T.; LAPIS, T.; SIT, S. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill Companies, Inc., 2012.